ＦＣＡＤ４１２ＰＣＩ－１６ＭＷ
取扱説明書

１６チャンネル逐次サンプリング

１６ＭＷバッファメモリ内蔵

１２ビット高速アナログ変換基板

対応パソコン
IBM PC/AT互換機 （PCI）

２０１５年3月６日　ドライバーアンインストール操作説明追加
２０１４年8月４日　Get_Status関数不具合修正
２０１４年6月３日　３２ビット版汎用IOハンドラー不具合修正
２０１４年4月１７日発行　初版

フレックスコア

http://www.flexcore.jp

合同会社　フレックスコア

目次

本製品の使用・適用についての注意事項 5
故障・修理・サポート方法について 6
本製品の構成 6
第１章　導入・試運転 8
１－１　本製品の概要 8
１－２　ボード上の設定 9
１－３　入出力コネクタ・ピン接続 10
アナログ入力　プラグ：１７ＪＥ－２３３７０－０２（Ｄ８Ａ）　ＤＤＫ製・添付 10
デジタルコネクタ 11
１－４　入出力接続オプション 11
１－５　ボードのインストール 11
１－６　動作確認・試運転 11
第２章　信号入出力 12
２－１　アナログ入力回路 12
２－２　アナログ入力範囲 12
２－３　アナログ入力特性（誤差、ドリフト、雑音、保護対策） 13
２－４　デジタル入出力回路 14
第３章　制御・操作 15
３－１　ＡＤサンプリング動作、トリガ動作の様子 15
有限自動サンプリングモード 15
無限自動サンプリングモード 15
ユーザーバッファ使用自動サンプリングモード 15
マニュアルサンプリングモード 15
３－２　オンボードメモリの構造、動作 15
３－３　制御レジスタアドレスマップ 16
３－４　ボード制御部リセット 17
３－５　アナログ回路接続設定・読出し 18
３－６　データコード指定 20
３－７　アナログ入力スキャン最終チャンネル指定 20
３－８　クロック源選択 20
３－９　クロック源分周比設定 21
３－10　アナログトリガレベル設定 21
３－11　トリガモード設定（ソフトトリガ実行を含む） 22
３－12　マニュアルサンプリングスキャン 22
３－13　ステータス入力操作 23
３－14　ステータスクリア操作 23
３－15　ＦＩＦＯメモリ保持データ数読込み 24
３－16　メモリサイズ判定 24
３－17　ＡＤデータ読出し 24
３－18　割り込み制御（要因設定） 24
３－19　割り込み要求クリア 25
３－20　汎用デジタル入出力 25
３－21　サンプルホールド回路制御（オプション） 26
３－22　マスタースレーブ接続説明（予約事項） 26
ボード上の設定 26
ボード間の接続 26

ソフトウェア設定 26
第４章　ソフトウェア 27
４－０　ハードウェアとソフトウェアの連携 27
４－１　本製品用ソフトウェアの種類 28
４－２　ドライバーのインストール 28
４－２－１　WindowsXP ドライバーインストール及びアンインストール 28
４－２－２　Windows 7 （X86：32 ビット）へのインストール 32
４－２－３　Windows 7 （X64：64 ビット）へのインストール 35
４－３　Windows 7 ドライバーのアンインストール 38
４－４　サンプルプログラムについて 40
~~４－５　サンプルプログラム、動作確認プログラムについて（補足）~~ 41
４－６　関数仕様、エラーコード 42
【１】ボード及び本ハンドラーの初期化 43
【２】サンプリング実行チャンネル関連の設定 43
【３】サンプリングモードの設定 43
【４】トリガ条件の設定 43
【５】レンジトリガ関連の設定 44
【６】オプション外部クロック値の設定 44
【７】クロック源及びクロック値の設定 44
【８】サンプリングスタート 44
【９】ボードステータスの取得 45
【10】本ハンドラーの終了 45
【11】DLL管理バッファからのデータ読出し 45
【12】ユーザーメモリへの直接データ転送 45
【13】マニュアルサンプリング 45
【14】汎用デジタル出力 46
【15】汎用デジタル入力 46
【16】リングバッファ指定有無 46
【17】アナログ入力モードの設定 46
【18】サンプリングの強制停止 46
【19】残りＡＤデータの読み込み 47
【20】各フラグのクリア 47
【21】外付け・同時サンプルホールド制御 47
【22】本ハンドラーのバージョン情報取得 47
FCAD412 独自機能用関数セット 47
【23】ハーフフルサイズの取得 48
【24】メモリ使用データ数の取得 48
【25】AD入力モードの設定 48
【26】AD入力モードの読出し 48
４－７　エラーコード 49
第５章　汎用デバイスドライバー仕様書 50
【１】ドライバーオープン 51
【２】ドライバースタート 51
【３】バイトリード 51
【４】バイトブロックリード 51
【５】ワードリード 52
【６】ワードブロックリード 52

【７】ダブルワードリード……52
【８】ダブルワードブロックリード……52
【９】バイトライト……53
【10】バイトブロックライト……53
【11】ワードライト……53
【12】ワードブロックライト……53
【13】ダブルワードライト……54
【14】ダブルワードブロックライト……54
【15】ドライバーストップ……54
【16】ドライバークローズ……54
エラーコード……55
第６章　保守、その他……56
６－１　故障・トラブル等の原因と対処……56
再点検、確認ポイント……56
ボードID……56
トリガ方法……56
デジタル入出力……56
アナログ入力……56
６－２　修理のときは……56
６－３　アナログ入力範囲の再調整……57
６－４　外部制御信号、タイミング等……57

本製品の使用・適用についての注意事項

（1）　本製品は、ＩＢＭＰＣ／ＡＴ互換機のＰＣＩバス拡張Ｉ／Ｏスロット、またはＰＣＩバス拡張Ｉ／Ｏボックスに装着して使用するものです。

（2）　本製品が組み込まれたシステムの運用対象・方法・場所・環境等によって、故障・誤動作等が生じた場合に起こり得る、身体・生命・財産等に対する損害の回避処置は同システムの設計・製作に別途付加・反映させて下さい。本製品自体には、前述の機能はなく、従って弊社では本製品が組み込まれたシステムの運用により発生した故障・誤動作・事故に起因する身体・生命・財産等の損害に対する責任は負えません。これは本製品の故障・誤動作が原因となった場合も含み、理由の如何を問いません。

（3）　本製品付属のソフトウェアは本製品利用の方法を示す例、またオプションの関連ソフトウェアは本製品利用の一般的便宜をはかるものであり、現在未発見のバグ存在の可能性も含めて、運用結果についての責任は一切負えません。
これらのソフトウェアには自身が組み込まれたシステムに故障・誤動作・事故等が生じた場合に起こり得る身体・生命・財産等に対する損害の回避機能はありません。御利用の場合は同システムの設計・製作で配慮・付加・反映させて下さい。

（4）　本製品（付属ソフトウェアを含む）、及びオプションの関連ソフトウェアは医用・航空機器用・その他高信頼性・高安全性を必要とするシステムに使用しないで下さい。

（5）　本製品付属のソフトウェアについて弊社は著作権を保持しますが、第３者の権利を侵害しない限りにおいて購入者は自身が製作するシステム等に自由に組み込み、販売する事もできます。但し、弊社製ソフトウェアのソースコードを含むソフトウェアを第３者に販売･移転するときは弊社の文書による事前許可を必要とします。

（6）　弊社では、本製品の販売・サポート・保証の範囲を日本国内に限っています。

（7）　本製品は改良のため仕様変更、価格改定を行うことがあります。

故障・修理・サポート方法について

（1）　納入後１年間は自然故障､及び弊社製造上の問題に起因した事が明らかな故障製品に対して無償修理を行います。

（2）　保証期間中であっても、次の場合は有償修理となります。

ア　取扱上の不注意、誤用による故障および損傷
イ　弊社以外での修理、改造、分解掃除等による故障および損傷
ウ　泥、砂、水などのかぶり、落下、衝撃等が原因で発生した故障および損傷
エ　火災、地震、水害、落雷その他の天変地異、公害や異常電圧による故障および損傷
オ　保管上の不備（高温多湿の場所等）や手入れの不備による故障
カ　接続している他の機器に起因して生じた故障
キ　その他使用者側の責に帰する故障

（3）　修理は宅配便によるセンドバックで行います。尚運賃は互いに発送する側が負担するものとします。
出張修理は行っておりません。簡単な故障であれば一週間程度で修理・返却が可能ですが、故障状況によっては更に日数を要します。

（4）　本製品使用上の質問・トラブル対応・故障修理などは入手経路の如何にかかわらず、弊社宛に直接御相談下さい。その際は、客観情報の整理・評価を行うために必ずレポートを御送付下さい。特にEmailで情報を頂く事で解決速度が大幅に向上する事が期待できます。（support@flexcore.jp）また、内容が複雑になっている場合はfax使用も有効です。

（5）　有償修理の場合の修理費用は、基本料金￥１０，０００円＋部品代となります。簡単な故障では凡そ２万円以下程度だと思われます。修理費用限度額がある場合は、お申し付け頂ければ超過する場合に御連絡致します。

（6）　修理品送付先

〒301-0853
茨城県龍ヶ崎市松ヶ丘３丁目１８番地３
フレックスコア
品質管理部　林
E-mail: support@flexcore.jp
Fax:050-3488-3354

本製品の構成

本製品は　ＦＣＡＤ４１２ＰＣＩ－１６MWボード
入出力プラグ（アナログ、デジタル各１個）
からなります。
ドライバーソフトウェア、取扱説明書（本文書）、サンプルプログラムなどについては、
弊社ホームページよりダウンロードをお願いします。

本機の仕様一覧

アナログ入力部

項目	ＦＣＡＤ４１２ＰＣＩ－１６MW
入力数・信号モード	１６チャンネルシングルエンドまたは８チャンネル差動入力
入力範囲 （半導体マルチプレクサ制御）	±１０Ｖ／±５Ｖ／±２．５Ｖ／０～＋１０Ｖ／０～＋５Ｖ 本機の入力範囲設定回路は高精度部品を使用しているため入力範囲を切り替えても通常の用途では再調整不要の誤差範囲に収まります。
入力インピーダンス	各チャンネル毎に１０MΩの終端抵抗を標準実装
クロストーク	－６５ｄＢ

ＡＤ変換部

項目	ＦＣＡＤ１０８ＳＰＣＩ－１６MW
分解能	１２ビット
単チャンネルサンプル速度	４ｕＳ（２５０KHｚ）
複数チャンネルサンプル速度	（８ｕＳ／Ｃｈ）
非直線性$_{max}$	±０．０２５%ＦＳ
正確度１	±０．０８７%ＦＳ
正確度２	±０．１０７%ＦＳ
内部雑音$_{typ}$	±１ＬＳＢ
温度ドリフト$_{typ}$	±２５ｐｐｍ／℃
ＡＤデータ・コード	バイナリまたは２の補数（ソフトウェア指定）

制御部等

項目	ＦＣＡＤ１０８ＳＰＣＩ－１６MW
クロック	クロック源：内部１０MHｚ／内部８．１９２MHｚ／外部 TTL 入力 分周（バイナリ）：３２ビットプログラマブルカウンタ
トリガ	内部トリガ：プログラム上からの即トリガ アナログ入力（先頭チャンネル）の指定エッジ、レベルまたはレンジ 外部トリガ：外部ＴＴＬ入力の指定エッジまたはレベル
バッファメモリ	１６M語メモリ（ＦＩＦＯモード）
ＡＤデータ転送	ブロック転送（データ長はプログラムにより取得可能）若しくはワード単位転送
マスタースレーブ動作	マスターのクロック出力をスレーブに接続する事により可能
割り込み	割り込み要因：１回サンプリングスキャン終了、トリガ発生 サンプリングクロック、外部ＴＴＬ入力の指定エッジ ＦＩＦＯメモリのハーフフル遷移
汎用デジタル入出力	１ビットＴＴＬ入力、１ビットＴＴＬ出力（またはオープンコレクタ出力）
制御アドレス	組み込み対象システムのプラグアンドプレイ機能により連続アドレスを占有
基板寸法	ＰＣＩショートサイズ（１７４．３mm×９８．４mm）突起部を含まず
動作環境	周囲温度０℃～４０℃（結露しない事）、保存温度―１０℃～+８０℃（結露しない事）
付属品	入出力プラグ
電流消費	＋５Ｖ，０．９Ａ

第１章　導入・試運転

１－１　本製品の概要

本ボードは、SDRAM を使用した大容量 FIFO メモリ搭載アナログサンプリング基板で、指定された条件（クロック、トリガ、チャンネル数）に従って、自動サンプリングを行う構造となっています。ソフトウェア上では、条件設定、サンプリングスタート（トリガ待ちを含む）ステータス評価／使用メモリ量確認／データ取り込みだけで、動作タイミングなどの制御が不要です。そのため、リアルタイム制御或いは Windows 環境でのデータ処理等に最適です。

- 高精度部品の使用によりアナログ入力範囲切換え時の再調整不要
- サンプリング点数を制限しない大容量 FIFO メモリ（１６M 語標準）
- 複数クロック源（内部１０MHｚ／内部８．１９２MHｚ／外部 TTL 入力）
- 複数トリガモード（ソフトウェア即トリガ／アナログトリガ／外部トリガ
- クロック入出力機能により複数ボードの同期運転可能

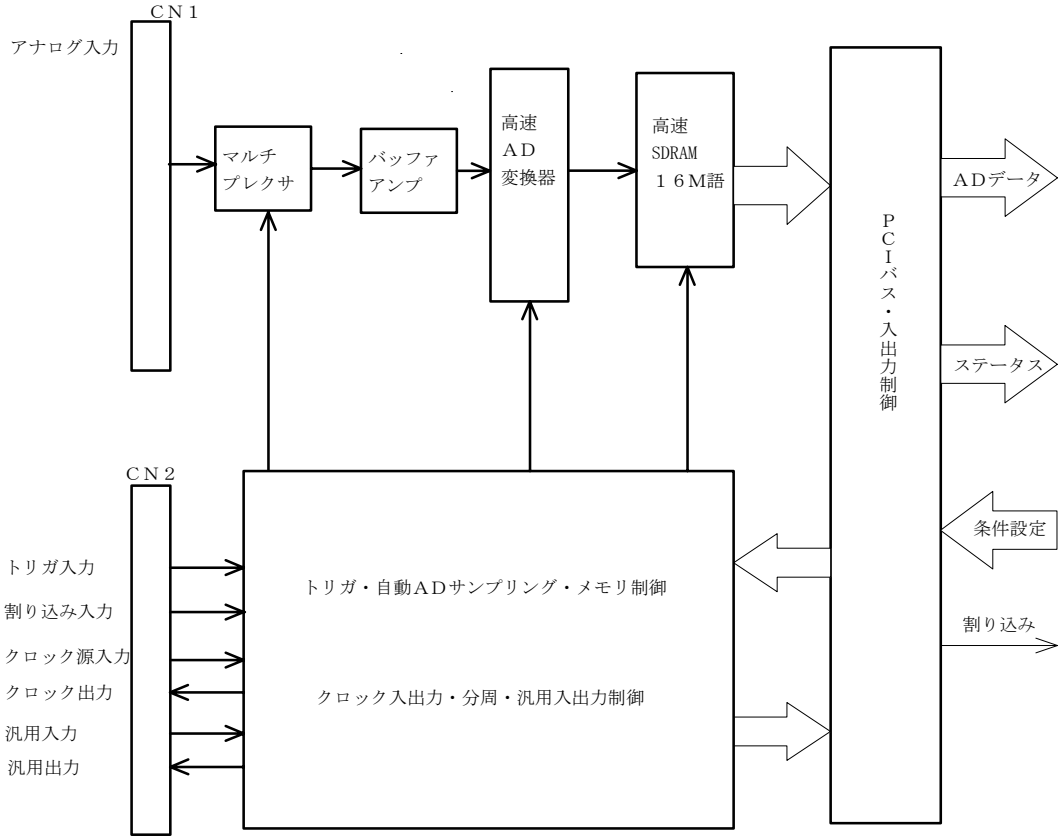


図１－１　ＦＣＡＤ４１２ＰＣＩ機能ブロック概念図

１－２　ボード上の設定

図１－２にＦＣＡＤ４１２ＰＣＩのボード上部品配置を示します。各機能については、同図下部の説明を参照してください。

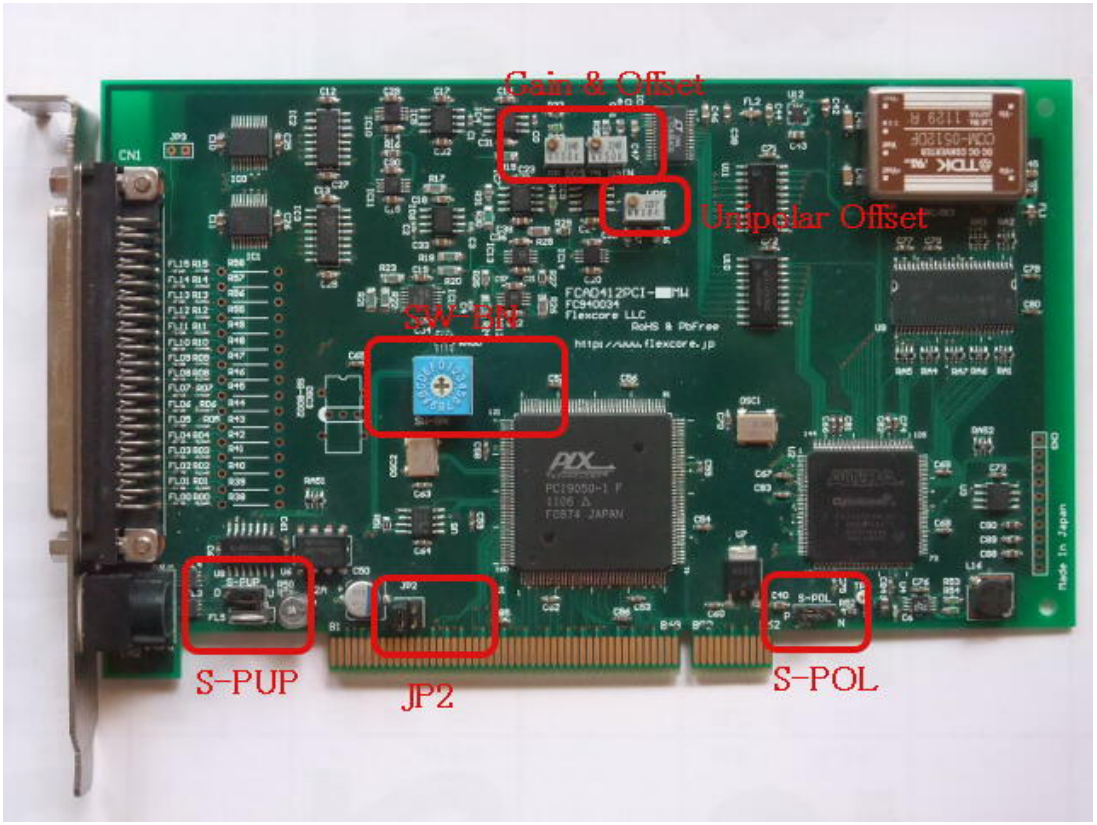


図１－２　ボード上の設定スイッチ等配置

SW-BN:ボード番号設定スイッチ（出荷時：０／複数ボード使用時のボード認識用）
アナログ入力スケール選択：MUX による電子回路切り替え機能（±１０Ｖ、±５Ｖ、±２．５Ｖ等）
アナログ入力範囲極性選択：MUX による電子回路切り替え機能（バイポーラ／ユニポーラ切り替え）
アナログ入力スパン切り替え選択：MUX による電子回路切り替え機能（Aモード／B モード）
S-POL:ジャンパープラグによる機能切り替え（デジタル出力の論理：正論理／負論理）
S-PUP:ジャンパープラグによる機能切り替え（デジタル出力の構造：プルアップ抵抗有無切り替え）
S-CKZ:外部クロック入力終端
TM0:オフセット調整トリマ
TM1:ゲイン調整トリマ
TM2:バッファオフセット調整トリマ

FS1:＋５Ｖ電源出力保護ヒューズ（FRPU-2.0A：浜井電球製）
CN1:アナログ入力コネクタ（３７ピン D-SUB）
CN2:デジタル入出力コネクタ（８ピン丸型コネクタ）

１－３　入出力コネクタ・ピン接続

アナログ入力には３７ピン D-SUB コネクタ、また汎用デジタル入出力を含む制御信号入出力には８ピン丸型コネクタを使用しており適合プラグ各１個が添付されています。

アナログ入力　プラグ：１７ＪＥ－２３３７０－０２（Ｄ８Ａ）　ＤＤＫ製・添付

基板側：１７ＬＥ－１３３７０－２７（Ｄ４ＡＢ）　ＤＤＫ製・実装

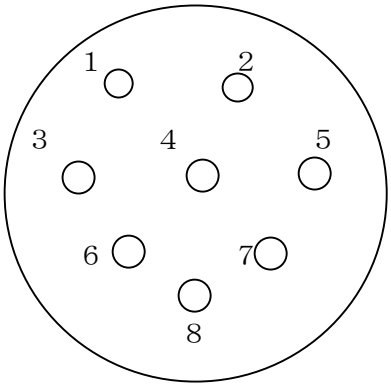
図１－３Ａ　アナログ入力コネクタ（ＣＮ１）ピン接続

信号名	機能	ピン番号		ピン番号	信号名（機能）
ＣＨ０（０Ｈ）	ＣＨ０入力（差動：ＣＨ０の＋側）	１	○		
				○ ２０	ＡＧ（アナロググランド）
ＣＨ１（０Ｌ）	ＣＨ１入力（差動：ＣＨ０のー側）	２	○		
				○ ２１	ＡＧ（アナロググランド）
ＣＨ２（１Ｈ）	ＣＨ２入力（差動：ＣＨ１の＋側）	３	○		
				○ ２２	ＡＧ（アナロググランド）
ＣＨ３（１Ｌ）	ＣＨ３入力（差動：ＣＨ１のー側）	４	○		
				○ ２３	ＡＧ（アナロググランド）
ＣＨ４（２Ｈ）	ＣＨ４入力（差動：ＣＨ２の＋側）	５	○		
				○ ２４	ＡＧ（アナロググランド）
ＣＨ５（２Ｌ）	ＣＨ５入力（差動：ＣＨ２のー側）	６	○		
				○ ２５	ＡＧ（アナロググランド）
ＣＨ６（３Ｈ）	ＣＨ６入力（差動：ＣＨ３の＋側）	７	○		
				○ ２６	ＡＧ（アナロググランド）
ＣＨ７（３Ｌ）	ＣＨ７入力（差動：ＣＨ３のー側）	８	○		
				○ ２７	ＡＧ（アナロググランド）
ＣＨ８（４Ｈ）	ＣＨ８入力（差動：ＣＨ４の＋側）	９	○		
				○ ２８	ＡＧ（アナロググランド）
ＣＨ９（４Ｌ）	ＣＨ９入力（差動：ＣＨ４のー側）	１０	○		
				○ ２９	ＡＧ（アナロググランド）
ＣＨ１０（５Ｈ）	ＣＨ１０入力（差動：ＣＨ５の＋側）	１１	○		
				○ ３０	ＡＧ（アナロググランド）
ＣＨ１１（５Ｌ）	ＣＨ１１入力（差動：ＣＨ５のー側）	１２	○		
				○ ３１	ＡＧ（アナロググランド）
ＣＨ１２（６Ｈ）	ＣＨ１２入力（差動：ＣＨ６の＋側）	１３	○		
				○ ３２	ＡＧ（アナロググランド）
ＣＨ１３（６Ｌ）	ＣＨ１３入力（差動：ＣＨ６のー側）	１４	○		
				○ ３３	ＡＧ（アナロググランド）
ＣＨ１４（７Ｈ）	ＣＨ１４入力（差動：ＣＨ７の＋側）	１５	○		
				○ ３４	ＡＧ（アナロググランド）
ＣＨ１５（７Ｌ）	ＣＨ１５入力（差動：ＣＨ７のー側）	１６	○		
				○ ３５	ＡＧ（アナロググランド）
	空きピン	１７	○		
				○ ３６	空きピン
	Ｓ／Ｈ	１８	○		
				○ ３７	ＤＧ（デジタルグランド）
	＋５Ｖ出力	１９	○		

注　アナロググランドとデジタルグランドはボード内部で接続されています。

デジタルコネクタ

図１－３Ｃ　デジタル入出力コネクタ・ピン接続



添付プラグの半田付け側面視

１ピン：割り込み入力
２ピン：トリガ入力
３ピン：クロック入力
４ピン：クロック出力
５ピン：汎用デジタル入力
６ピン：汎用デジタル出力
７ピン：＋５Ｖ出力
８ピン：デジタルグランド

１－４　入出力接続オプション

本製品を含む３７ピンＤ－ＳＵＢコネクタを使用したアナログ変換基板共通のＢＮＣ端子ボックス（ＣＢＯＸ－０１４　１６Ｃｈ）が用意されています。本ボックスは、アナログ変換基板と２本の１m長ケーブルで接続され、アナログ入力をＢＮＣ端子で、デジタル入出力をＢＮＣ端子／端子台で処理する事ができるため、信号の取り回しが容易です。

１－５　ボードのインストール

本製品はプラグアンドプレイに対応した PCI ボードです。そのため御使用に先立ち組み込むパソコンにインストール（認識、リソース割当て）する必要があります。そのためのステップは次の通りです。

１　まず、弊社ホームページよりドライバーパッケージをダウンロードします。そして、このパッケージを展開解凍し、その後インストール作業を行います。この際、本ボードはまだパソコンに取り付けてはいけません。

２　デバイスドライバーをインストールした後、一旦パソコンの電源を落とし、その後本ボードを PCI スロットに取り付けてからパソコンの電源を再投入します。一旦システムトレイにドライバーインストール中のメッセージが表示され、一分ほど待っていると、ドライバーがインストールされ、ハードウェアの使用準備ができた旨、表示されます。

３　この段階で本ボードを使用する事ができる状態になっていますので、必要に応じ適宜弊社ホームページよりサンプルプログラムをダウンロードして頂き実行モジュールを作成するか、或いはご自身で使用される言語システムを利用した開発を開始する事ができます。より詳細な内容については４－１章を参照して下さい。

１－６　動作確認・試運転

Windows7 対応の動作確認プログラム並びにサンプルプログラム（Ｃ＋、ＶＢ６等）を用意しています。詳細は弊社ホームページよりダウンロードできるソフトウェアキットを参照してください。

第２章　信号入出力

２－１　アナログ入力回路

アナログ入力は、マルチプレクサで順次切り替えＡＤ変換される構造です。

アナログ入力範囲はボード上のＭＵＸ（ＭＵＸ１～ＭＵＸ３）によって選択され、絶対最大定格は±４０Ｖとなっています。

２－２　アナログ入力範囲

アナログ入力範囲は、ボード内ＭＵＸによって表２－２Ａの中から選択します。１２ビット分解能は、１／４０９６ですが、公称入力範囲で正直に調整するとＡＤ変換値１デジット当たりの電圧値が割り切れない値となります。そこで、Ａモードと称し、公称入力範囲を４０００で除した分解能を採用したモードを準備しています。

表２－２Ａ　アナログ入力電圧入力範囲・分解能・正確度

公称入力範囲	範囲モード	実際の入力範囲 （±１０Ｖを越えた部分は無効）	分解能 mV/digit	正確度
±１０Ｖ	Ａ	－１０．２４Ｖ～＋１０．２３５Ｖ	５	表２－２Ｃ１
	Ｂ	－１０．０００Ｖ～＋９．９９５１２Ｖ	４．８８	
±５Ｖ	Ａ	－５．１２Ｖ～＋５．１１７５Ｖ	２．５	
	Ｂ	－５．０００Ｖ～＋４．９９７５６Ｖ	２．４４	
±２．５Ｖ	Ａ	－２．５６０Ｖ～＋２．５５８７５Ｖ	１．２５	
	Ｂ	－２．５００Ｖ～＋２．４９８７８Ｖ	１．２２	
０～＋１０Ｖ	Ａ	０Ｖ～＋１０．２３７５Ｖ	２．５	
	Ｂ	０Ｖ～＋９．９９７５６Ｖ	２．４４	
０～＋５Ｖ	Ａ	０Ｖ～＋５．１１８７５Ｖ	１．２５	
	Ｂ	０Ｖ～＋４．９９８７８Ｖ	１．２２	

表２－２Ｂ及び表２－２ＣにＡモード及びＢモード設定夫々の場合について主要なアナログ入力電圧と変換デジタルデータの関係を纏めています。

表２－２Ｂ　１２ビットＡＤデータｖｓアナログ入力（Ａモード）

ＡＤデータ	アナログ入力範囲					
Hex/10進	±１０Ｖ	±５Ｖ	±２．５Ｖ	０～＋１０Ｖ	０～＋５Ｖ	０～＋２．５Ｖ
FFF/4095	+10.235	+5.1175	+2.55875	+10.2375	+5.11875	+2.55938
FD0/4048	+10.000	+5.0000	+2.5000			
FA0/4000				+10.0000	+5.0000	+2.50000
801/2049	+0.005	+0.0025	+0.00125			
800/2048	0.000	0.0000	0.00000			
7FF/2047	-0.005	-0.0025	-0.00125			
7D0/2000				+5.0000	+2.50000	+1.25000
030/48	-10.000	-5.0000	-2.50000			
001/1	-10.235	-5.1175	-2.55875	+0.0025	+0.00125	+0.000625
000/0	-10.24	-5.1200	-2.56000	0.0000	0.00000	0.00000

表２－２Ｃ　１２ビットＡＤデータｖｓアナログ入力（Ｂモード）

ＡＤデータ	アナログ入力範囲					
Hex/10 進	±１０Ｖ	±５Ｖ	±２．５Ｖ	０～＋１０Ｖ	０～＋５Ｖ	０～＋２．５Ｖ
FFF/4095	+9.99512	+4.99756	+2.49878	+9.99756	+4.99878	+2.49939
FD0/4048	+9.76563	+4.88281	+2.44141			
FA0/4000				+9.76563	+4.88281	+2.44141
801/2049	+0.00488	+0.00244	+0.00122			
800/2048	0.00000	0.0000	0.00000	+5.00000	+2.50000	+1.25000
7FF/2047	-0.00488	-0.00244	-0.00122			
7D0/2000				+4.88281	+2.44141	+1.22070
030/48	-9.76563	-4.88281	-2.44141			
001/1	-9.99512	-4.99756	-2.49878	+0.00244	+0.00122	+0.00061
000/0	-10.00000	-5.00000	-2.50000	0.00000	0.00000	0.00000

２－３　アナログ入力特性（誤差、ドリフト、雑音、保護対策）

ＡＤ変換誤差：　本ボードのＡＤ入力は±１０Ｖ範囲で調整されていますが、高精度部品の使用により入力範囲を切り替えても多くの用途では再調整の必要がありません。しかしながら特定の入力範囲で最も正確度を良くしたいときは再調整が必要です。
温度ドリフト：　ＡＤボードの周囲温度が変化した時、同一アナログ入力に対するＡＤ変換データが変化する度合いを示します。本ボードでは２５ｐｐｍ／℃typ となります。
経年変化：　経年変化のデータはありません。充分な精度の維持が必要な用途では年に１～２回程度標準電圧源などで校正し必要に応じて再調整を行うと良いと思われます。

内部雑音：　本ボードでは内部雑音は１２ビット設定時±１LSB,１４ビット設定時は±４LSB となります。

入力耐圧：　本ボードでは±４０Ｖまでの過電圧に対して保護されていますが、これを超える入力電圧が印加されると入力回路素子の破壊による故障の原因となります。このような場合は、抵抗＋ツェナーダイオード等の保護回路が有効です。実際の定数を決定するには、ある程度のカットアンドトライが必要になると思われます。

２－４　デジタル入出力回路

外部トリガ入力、外部割込み入力、汎用１ビットデジタル入力は全てＴＴＬレベルであり、夫々１０ＫΩで、プルアップされています。クロック出力もＴＴＬレベルです。

汎用１ビットデジタル出力はオープンコレクタですが、接続先がＴＴＬレベルの時は、ボード上のジャンパーＳ－ＰＵＰにより１ＫΩのプルアップ抵抗を接続して使用する事ができます。（出荷時設定）

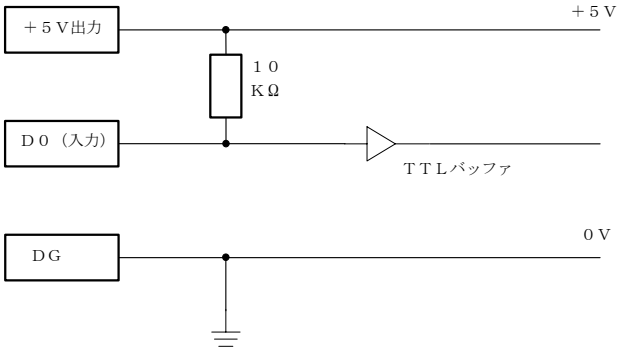


図２－４Ａデジタル入力

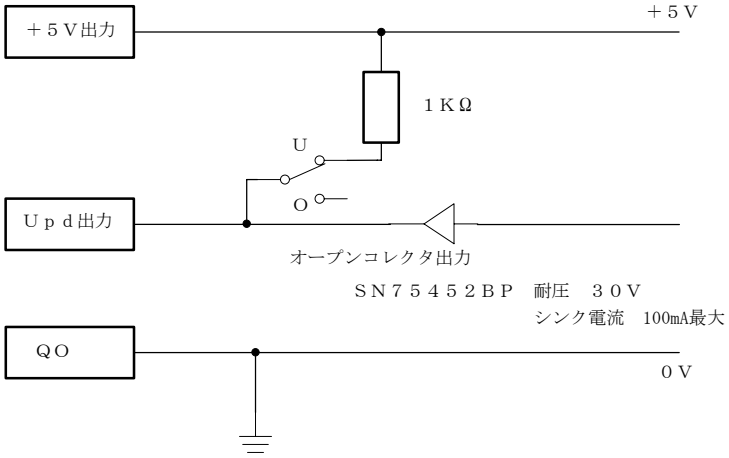


図２－４Ｂ　汎用１ビット出力

第３章　制御・操作

３－１　ＡＤサンプリング動作、トリガ動作の様子

本ボードには大別して２種類のサンプリングモードがあります。

第一に挙げるモードは自動サンプリングモードです。このモードは、予め設定されたチャンネル数、トリガモード等によってボード内部で自動的にサンプリングを継続実行するモードです。また、このモードではドライバー内部で使用する DLL バッファの使用法により有限サンプリングモードと、リングバッファとして使用する無限サンプリングモード及びユーザー定義のバッファへサンプリングデータを保存する３種類の動作モードがあります。

最後に挙げるモードはマニュアルサンプリングモードです。このモードでは、あらかじめ設定されたチャンネル群に対し、１サイクルのサンプリングを行います。

有限自動サンプリングモード

このモードでは、ドライバー内部の DLL バッファをサンプリング開始に先立ち必要な容量分確保しておきこの状態でサンプリングが開始されると、サンプリング数が予めセットされた値になった事を確認しサンプリング動作が停止されます。

無限自動サンプリングモード

このモードでは、有限自動サンプリングモードと同様、ドライバー内部に DLL バッファを確保しますが、サンプリング開始されるとそのデータを上記バッファにリング状に保存して行きます。そのため、この場合にサンプリングを停止するためには、明示的にサンプリングストップを指示する必要があります。

ユーザーバッファ使用自動サンプリングモード

このモードでは、サンプリングデータ保存用バッファとしてドライバー内部には領域の確保を行わず、ユーザー定義のバッファへサンプリングデータを取り込みます。そのために本ドライバー側からユーザーアプリケーション側へメッセージ送信が行われ、ユーザーアプリケーション側では、このメッセージ及びステータス関数によって得られるボードのステータスから取り込み可能なデータ数をアプリケーション側のバッファへ転送する処理を行います。尚、メッセージとして処理されるボード側の条件はハーフフル条件成立またはノットエンプティ条件成立の何れかになります。

マニュアルサンプリングモード

このモードでは、サンプリングパラメータを設定し、その後ソフトウェア上でマニュアルサンプリング開始コマンドを発行することで、１サイクルのサンプリングを行います。このモードは、通常、サンプリング動作のデバッグモード或いは、サンプリング動作そのものに時間的要因が含まれない動作の場合に使用されます。

３－２　オンボードメモリの構造、動作

本ボードには、通常の FIFO メモリではなく汎用の SDRAM を FPGA で制御する事による擬似的な FIFO メモリが実装されています。この事により

１　通常の FIFO メモリに対しはるかに大容量のメモリを使用できる

２　FIFO メモリ内部動作が FPGA によって実現されているため、この擬似メモリの内部動作が本ボード経由でアプリケーション側からも把握でき、その結果より細かい制御が可能となる

というような利点を挙げる事ができます。

３－３　制御レジスタアドレスマップ

表３－３に本ボードのアドレスマップを示します。
表中【ＢＡＳＥ】で示される数値は、プラグアンドプレイで割り付けられるＩＯアドレスのベース値です。

表３－３　制御レジスタのＩ／Ｏアドレス

Ｉ／Ｏアドレス(Hex)	ＩＮ／ＯＵＴ	ポート／レジスタ名・機能	参照
【ＢＡＳＥ】＋Ｆ	ＯＵＴ		
	ＩＮ		
【ＢＡＳＥ】＋Ｅ	ＯＵＴ		
	ＩＮ		
【ＢＡＳＥ】＋Ｄ	ＯＵＴ		
	ＩＮ		
【ＢＡＳＥ】＋Ｃ	ＯＵＴ	外付け同時サンプルホールド制御（オプション）	３－21 項
	ＩＮ	アナログ回路設定状態読出し（注１）	３－５項
【ＢＡＳＥ】＋Ｂ	ＯＵＴ	アナログ回路接続設定、ハーフフルサイズ設定（注１）	３－５項
	ＩＮ	ボード番号（ロータリースイッチ設定値）	３－４項
【ＢＡＳＥ】＋Ａ	ＯＵＴ	汎用デジタル出力（ラッチ）	３－20 項
	ＩＮ	汎用デジタル入力（現在値）	同上
【ＢＡＳＥ】＋９	ＯＵＴ	クロック源分周比設定	３－９項
	ＩＮ	メモリサイズコーディング情報	３－16 項
【ＢＡＳＥ】＋８	ＯＵＴ	クロック源選択	３－８項
	ＩＮ	ＦＩＦＯメモリ保持データ数読込み（注１）	３－15 項
【ＢＡＳＥ】＋７	ＯＵＴ	アナログトリガレベル（２）設定	３－10 項
	ＩＮ	ボード制御部リセット	３－４項
【ＢＡＳＥ】＋６	ＯＵＴ	アナログトリガレベル（１）設定	３－10 項
	ＩＮ	マニュアルサンプリングスキャン	３－12 項
【ＢＡＳＥ】＋５	ＯＵＴ	ステータス（ビット指定）クリア	３－13 項
	ＩＮ	ステータス取得	３－14 項
【ＢＡＳＥ】＋４	ＯＵＴ	割り込み制御（要因設定）	３－18 項
	ＩＮ		
【ＢＡＳＥ】＋３	ＯＵＴ	トリガモード指定（ソフトトリガ実行を含む）	３－11 項
	ＩＮ		
【ＢＡＳＥ】＋２	ＯＵＴ	データコード指定	３－６項
	ＩＮ		
【ＢＡＳＥ】＋１	ＯＵＴ	割り込み要求クリア	３－19 項
	ＩＮ		
【ＢＡＳＥ】＋０	ＯＵＴ	アナログ入力スキャン最終チャンネル指定	３－７項
	ＩＮ	ＡＤデータ読出し	３－17 項

ＩＮ／ＯＵＴはパソコン側から見た方向
全てのポートはバイトアクセス
（注１）：これらの機能は、FCAD412PCI で追加導入されている為、マイクロサイエンス製の専用ドライバー（H682_2K.Dll）では使用できません。（汎用 IO ドライバーでは使用可能です）

３－４　ボード制御部リセット

ｒｓｔ＝ｉｎｐ（ＢＡＳＥ＋０ｘ７）；　　//　制御部リセット操作

本ボード制御部全体を初期化します。この操作で読み込まれた値は本ボードのＩＤです。本操作は電源ＯＮ又はパソコン本体のハードウェアリセットと同等の機能ですが、汎用デジタル出力だけは変化せず保持されます。
本操作により

- 本ボード上の各制御レジスタを初期化する。
- ボードステータスを初期化する。
- サンプリング中であればこれを中止する。
- ＦＩＦＯメモリをクリアする。（格納された読出し待ちＡＤデータは失われる）
- アナログ回路設定情報を初期化する

尚、クロック源及び分周比は無効となるため再設定が必要となります。

表３－４Ａ【ＢＡＳＥ】＋０ｘ７Ｈ入力ポートの構成

ビット	各ビットの機能・意味
Ｂ７～Ｂ０	０ｘ０ＥＨ

【※】ここで読み込まれるボードＩＤはＰＣＩバス上のデバイスＩＤとは全く無関係です。

本ボードを複数使用する場合
本ボードのＩＯアドレスはプラグアンドプレイによりダイナミックに割り当てられる為、複数のボードを使用する場合ソフトウェア側から夫々のボードを特定する手段が必要になります。（この手段がないと、例えばハードウェア構成が変化した場合など、ソフトウェア側からボードの特定を行う事が困難になります。）
そのために、このスイッチが用意されています。
また、専用ハンドラーを使用する際には、例えボード１枚のみの使用であってもこのスイッチを“０”に設定しておかなければなりません。

ＢＮ＝ｉｎｐ（ＢＡＳＥ＋０ｘ０Ｂ）；　　//　ボード番号設定スイッチの値を読み込む

表３－４Ｂ【ＢＡＳＥ】＋０ｘ０ＢＨ入力ポートの構成

ビット	各ビットの機能・意味
B7～B4	未使用
B3 B2 B1 B0	ボード番号設定スイッチの値（０～ＦＨ）

注】　マイクロサイエンス製ドライバーでは複数枚仕様が組み込まれていないため、この仕様については単なるハードウェアの説明となっています。複数枚仕様については、将来的に関数セットを拡張する予定です。

３－５　アナログ回路接続設定・読出し

本ボードでは、ボード内部のアナログ回路設定を半導体スイッチによりソフトウェア的に切り替える事ができます。

ｒｅｔ＝ｉｎｐ（ＢＡＳＥ＋０ｘ０C）；　　// 設定値読み込みの場合

ｏｕｔｐ（ＢＡＳＥ＋０ｘ０B、Ｓｅｔ＿Ｄａｔａ＋Ｆｉｆｏ＿Ｓｉｚｅ）；// 設定値書き込みの場合

（Ｓｅｔ＿Ｄａｔａはレジスタに書き込む内容、Ｆｉｆｏ＿Ｓｉｚｅはハーフフルサイズ設定データ）

切り替えられる機能は、以下の通りとなります。

- 入力レンジ（２．５Ｖ、５Ｖ、１０Ｖ）
- 感度切り替え（Ａレンジ／Ｂレンジ）
- 入力モード切り替え（バイポーラ／ユニポーラ）
- 差動／シングルエンド入力切り替え
- ＦＩＦＯメモリとしてのハーフフルサイズ設定値

これらについて実際の組み合わせを表３－５Ａ～Ｂに示します。

入力レンジ	Ａレンジ	入力感度	Ｂレンジ	入力感度
±２．５Ｖ	-2.56V～+2.55875V	1.25mV/Digit	-2.5V～+2.49878V	1.22mV/Digit
±５Ｖ	-5.12V～5.1175V	2.5mV/Digit	-5V～+4.99756V	2.44mV/Digit
±１０Ｖ	-10.24V～10.235V	5mV/Digit	-10V～+9.99512V	4.88mV/Digit
０～＋２．５Ｖ	0～+2.55938V	0.625mV/Digit	0～+2.49939V	0.61mV/Digit
０～＋５Ｖ	0～+5.11875V	1.25mV/Digit	0～+4.99878V	1.22mV/Digit
０～＋１０Ｖ	0～+10.2375V	2.5mV/Digit	0～+9.99756V	2.44mV/Digit

表３－５Ａ　アナログ回路接続設定組み合わせ

表３－５Ｂには実際の回路との間で取り決められている出力ビット割り当てを示します。

ビット	機能	“1”の時	“0”の時
B7～B5	ハーフフルサイズ設定	B7～B5=０	２５６語
		B7～B5=１	５１２語
		B7～B5=２	１０２４語
		B7～B5=３	２０４８語
		B7～B5=４	４０９６語
		B7～B5=５	８１９２語
		B7～B5=６	１６３８４語
		B7～B5=７	３２７６８語
B4	差動／シングルエンド	シングルエンド	差動入力
B3,B2	入力感度切り替え	B3B2=00	２．５Ｖ
		B3B2=01	５Ｖ
		B3B2=10	１０Ｖ
		B3B2=11	使用不可
B1	ユニポーラ／バイポーラ切り替え	ユニポーラ	バイポーラ
B0	A レンジ／B レンジ切り替え	B レンジ	A レンジ

表３－５Ｂ　【ＢＡＳＥ＋０ＢＨ】出力ポートの構成

表３－５Ｃでは、回路設定条件読み出しポートの入力ビット割り当てを示します。

ビット	機能	“1”の時	“0”の時
B7～B5	ハーフフルサイズ設定	B7～B5=0	２５６語
		B7～B5=1	５１２語
		B7～B5=2	１０２４語
		B7～B5=3	２０４８語
		B7～B5=4	４０９６語
		B7～B5=5	８１９２語
		B7～B5=6	１６３８４語
		B7～B5=7	３２７６８語
B4	差動／シングルエンド	シングルエンド	差動入力
B3,B2	入力感度切り替え	B3B2=00	２．５Ｖ
		B3B2=01	５Ｖ
		B3B2=10	１０Ｖ
		B3B2=11	使用不可
B1	ユニポーラ／バイポーラ切り替え	ユニポーラ	バイポーラ
B0	Ａレンジ／Ｂレンジ切り替え	Ｂレンジ	Ａレンジ

表３－５Ｄ　【ＢＡＳＥ＋ＣＨ】入力ポートの構成

また、以下に実際の例を示します。
　ハーフフルサイズ＝１０２４語
　シングルエンド入力
　入力レンジ：±５Ｖ
　ユニポーラ
　レンジ指定：Ｂレンジ
の場合、
　B7～5=010
　B4,=1
　B3,B2=01
　B1=1
　B0=1 となるので、これらを組み合わせた値、0x57 がこのレジスタに書き込まれるべき数値ということになります。しかし、実際には予めご使用いただく設定がハードコーディングされておりますので、この数値を直接出力しても所要のセッティングにはなりません。また、通常使用される場合にはこのレジスタを書き換える可能性はかなり低いものと思われます。実際に内容を変更するには、以下の手続きにより行います。

```
1　outp（BASE+0x0B,0）；　　// まず、このレジスタの内容を初期化します。
2　retc =(BASE+0x0C);　　// 初期状態に戻したレジスタの内容を取得します。
3　retc = retc^OutPutData;　　// 出力すべきデータとの間で排他的論理和を計算します。
4　outp (BASE+0x0B,retc);　　// 計算結果をレジスタに書き込みます。
```

特に、初期の値以外の設定がされている場合、ステップ１のレジスタ初期化を行わないと所要の設定ができないため注意が必要です。

３－６　データコード指定

ｏｕｔｐ（ＢＡＳＥ＋０ｘ２、ｄｃｄ）；

AD データコード（バイナリ、または２の補数）の設定データを書き込みます。

ビット	各ビットの機能・意味	＝１の時	＝０の時	リセット時
B7～B5	未使用			
B4	AD データコード	２の補数	バイナリ	0
B3～B0	未使用			

表３－６Ａ　【ＢＡＳＥ＋２H】出力ポートの構成

３－７　アナログ入力スキャン最終チャンネル指定

ｏｕｔｐ（ＢＡＳＥ＋０ｘ０、ｅｎｄ＿ｃｈ）；

サンプリングスキャン動作は、チャンネル０から、上の式で示される最終チャンネルまで連続して行います。
即ち、サンプリングチャンネル数はｅｎｄ＿ｃｈ＋１となります。

ビット	各ビットの機能・意味	設定値	リセット時
B7～B4	未使用		
B3～B0	スキャン最終チャンネル	０H～ＦH	0

表３－７Ａ　【ＢＡＳＥ＋０H】出力ポートの構成

３－８　クロック源選択

ｏｕｔｐ（ＢＡＳＥ＋０ｘ０８、ｃｌｋ）；　// ｃｌｋ：クロック源選択データ

ここで選択したクロック源信号は次項で説明するプログラマブルカウンタにより分周されて所要のサンプリングクロックになります。例えば、外部クロック源入力を選択し、分周比を１に設定すれば、外部イベントに同期したサンプリングを行うことができます。

ビット	各ビットの機能・意味	＝１の時	＝０の時	リセット時
B7	外部クロック源の有効極性指定	↑（+）	↓（−）	0
B6～B5	内部クロック源の選択データ	B6B5=11	無効	００
		B6B5=10	オプション	
		B6B5=01	８．１９２MHｚ	
		B6B5=00	１０．０００MHｚ	
B4	クロック源選択（外部／内部）	外部	内部	0
B3～B0	未使用			

表３－８　【ＢＡＳＥ＋０８H】出力ポートの構成

オプションのクロックは本基板上 OSC4 に追加装着することで使用可能となります。（１０MHｚ以下）
対応可能発振器は SG-8002DC-□□□M-PTBS 型（セイコーエプソン）が入手可能です。

３－９　クロック源分周比設定

ｏｕｔｐ（ＢＡＳＥ＋０ｘ０９、ＤｉｖＬＬ）；　// 分周比（下位バイトA）
ｏｕｔｐ（ＢＡＳＥ＋０ｘ０９、ＤｉｖＭＬ）；　// 分周比（上位バイトA）
ｏｕｔｐ（ＢＡＳＥ＋０ｘ０９、ＤｉｖＭＨ）；　// 分周比（下位バイトB）
ｏｕｔｐ（ＢＡＳＥ＋０ｘ０９、ＤｉｖＨＨ）；　// 分周比（上位バイトB）

クロック源の分周比を所要のレジスタに書き込みます。必ず連続して４バイトを書き込む必要があります。
実際の分周は１６ビットカウンターA，Ｂを直列に接続したカウンターによって行われます。
内部クロック源１０ＭＨｚを選択した時の最長クロック周期は
１００ｎＳ×０ｘＦＦＦＦ×０ｘＦＦＦＦ＝１００ｎＳ×４２９４８３６２２５＝４２９．４８３６２２５秒
となります。

３－10　アナログトリガレベル設定

ｏｕｔｐ（ＢＡＳＥ＋０ｘＣ、ＴｒｇＬ）；　// ＴｒｇＬ＝トリガレベルデータ１
ｏｕｔｐ（ＢＡＳＥ＋０ｘＥ、ＴｒｇＨ）；　// ＴｒｇＨ＝トリガレベルデータ２

アナログトリガレベル指定データをそれぞれのアドレスに書き込みます。各データはバイト長で、必ず両方のデータを書き込む必要があります。
（トリガモードによってはＴｒｇＨがダミーデータとなりますが書き込み操作そのものは必要です。）
アナログトリガ待ち状態では、チャンネル０入力が 4uS 周期で連続して監視サンプリングされ AD 変換データの上位８ビットが上記で設定されたトリガレベルと比較されています。トリガが認識されると３－１０項で示した分周動作が開始され、連続サンプリングが始まります。
ここで両トリガレベルと次項で説明するトリガモードとの間には、表３－１０Ａで説明される関係があります。

トリガレベルデータ	エッジトリガの場合	レベルトリガの場合	レンジトリガの場合
ＴｒｇＬ	トリガレベル(注)	トリガレベル	トリガレベル下限値
ＴｒｇＨ	トリガレベル(注)	ダミーデータ	トリガレベル上限値

表３－１０Ａ　トリガモード対トリガレベルデータ

注　エッジトリガの場合、ＴｒｇＨ－ＴｒｇＬの値がヒステリシスとなります。即ち立ち上がりエッジの場合はＴｒｇＬ未満からＴｒｇＨを超えた時点で、また立ち下がりエッジの場合はＴｒｇＨ以上からＴｒｇＬを下回った時点でトリガ成立となります。

アナログ入力範囲	Ａレンジの場合		Ｂレンジの場合	
	Vspan	分解能（Vspan/256）	Vspan	分解能（Vspan/256）
±10V	20.48V	80mV	20V	78.125mV
±5V、0～+10V	10.24V	40mV	10V	39.0625mV
±2.5V、0～+5V	5.12V	20mV	5V	19.5312mV
0～+2.5V	2.56V	10mV	2.5V	9.7656mV

表３－１０Ｂ　各アナログ入力範囲に対するスパン、トリガレベル分解能

３－11　トリガモード設定（ソフトトリガ実行を含む）

ｏｕｔｐ（ＢＡＳＥ＋０ｘ３、TrgMode）；
トリガモード及びサンプリングモードデータを書き込みます。

ビット	各ビットの機能・意味	＝１の時	＝０の時	リセット時
B7	ソフトトリガ制御	発生	禁止（強制停止）	0
B6	外部トリガ入力信号制御	許可	禁止（強制停止）	0
B5	内部（アナログ）トリガ制御	許可	禁止（強制停止）	0
B4	トリガ極性選択	↑（＋）	↓（－）	0
B3	トリガ認識モード選択２	エッジ	レベル	0
B2	トリガ認識選択モード１	レンジ	レンジ以外	0
B1	未使用			0
B0	サンプリングモード選択	連続サンプリング	マニュアルサンプリング	0

表３－１１【ＢＡＳＥ＋０ｘ３】出力ポートの構成

B7：ソフトトリガ制御ビットです。このビットをセットすると連続サンプリング開始と同時にサンプリングが開始されます。
B6：外部トリガ入力信号をサンプリングトリガとして使用するかどうかを指定するビットです。このビットをセットすると外部トリガ入力信号が B4,B3 で指定したモード条件に一致した時点でサンプリングが開始されます。
B5：アナログトリガ信号をサンプリング開始条件とするかどうかを指定するビットです。このビットをセットすると、アナログ入力信号が B4,B3 で指定される条件になった時点でサンプリングが開始されます。この条件には前項で説明したトリガレベルも関係します。
B4：B6,B5 で指定するトリガ信号の有効エッジ若しくは有効レベルについて立ち上がりエッジ／立下りエッジ若しくはハイレベル／ローレベルいずれを選択するかを指定するビットです。
B3：B6,B5 で指定するトリガ信号についてエッジ検出かレベル検出かを指定するビットです。
B2：B6,B5 で指定するトリガ信号についてその扱いをレンジモードとするかどうかを指定するビットです。
B0：サンプリング動作が連続サンプリングかマニュアルサンプリングかを指定するビットです。

３－12　マニュアルサンプリングスキャン

ｒｅｔ＝ｉｎｐ（ＢＡＳＥ＋０ｘ６）；　// ｒｅｔ：ダミーデータ
本操作により予め設定されたチャンネル群に対し、１サイクルのサンプリングを行います。変数ｒｅｔには特に意味は無く、この操作を行ったときのボード内部の動作によりサンプリングを行います。
サンプリングされたアナログデータは、通常のサンプリングと同様 FIFO メモリに書き込まれますので、アプリケーション側からは、同じように FIFO メモリからの読出しを行うことでアナログデータを得る事ができます。

３－13　ステータス入力操作

ｓｔｓ＝ｉｎｐ（ＢＡＳＥ＋０ｘ５）；
本ボードのステータスデータを読み込みます。

ビット	各ビットの機能・意味	＝１の時	＝０の時	リセット時
B7	EOS：各回サンプリングスキャン終了【注１】	終了済み	実行中、他	0
B6	INT：割り込み要求発生【注１】	発生済み	未発生、他	0
B5	TIM：各回サンプリングスキャン開始【注１】	開始済み	未開始、他	0
B4	TGD：トリガ発生認識【注２】	発生済み	未発生	0
B3	未使用			
B2	ERR：エラーフラグ【注１】	発生	未発生	0
B1	HLF：Not Half Full【注２】	１／２未満	１／２以上	1
B0	EMP：Not Empty【注２】	データあり	データなし	0

表３－１３Ａ【ＢＡＳＥ＋AH】入力ポートの構成

【注１】一旦セットされるとクリアされるまで状態を保持するラッチフラグ
【注２】現在の状態を常に反映する状態フラグ

各ビットの機能・意味詳細説明

EOS：　各回のサンプリングスキャンが終了した時点でセットされるフラグ
INT：　所要の割り込み条件が成立し、割り込み信号が出力された時点でセットされるフラグ
TIM：　各回のサンプリングスキャンが開始される度に、セットされるフラグ
TGD：　所要のトリガ条件が成立するとセットされる。またトリガ禁止操作でクリアされる。
ERR：　サンプリング動作時にエラーが発生するとセットされるフラグ、制御部クリアまで保持される。
　セット要因は
　１　FIFO メモリが満杯の状態で更に書き込みが行われた場合（データロストエラー）
　２　最高サンプリング速度を上回る速度でサンプリングが行われた場合（オーバーランエラー）
HLF：　FIFO メモリに対する所要データ数の書き込みが行われた時点でクリアされるフラグ
　このフラグがクリアされていれば、予めセットされたデータ数が FIFO メモリに格納されている事が保証されている。
EMP：　FIFO メモリに１バイト以上のデータがある時にセットされるフラグ、アプリケーション上は低速でサンプリングを行うケース以外ではほとんど使用されることはない。

３－14　ステータスクリア操作

ｏｕｔｐｗ（ＢＡＳＥ＋０ｘＡ、ＳｔａｔｕｓＣｌｅａｒ）；　// StatusClear はレジスタクリアパターン
ステータスレジスタの各ビットをクリアします。クリアするビットとの対応は表３-13Ｂによります。

ビット	各ビットの機能・意味	＝１の時	＝０の時	リセット時
B7	EOS：【注１】ビットのクリア	クリアする	クリアしない	0
B6	INT：【注１】ビットのクリア	クリアする	クリアしない	0
B5	TIM：【注１】ビットのクリア	クリアする	クリアしない	0
B4				
B3				
B2	ERR：【注１】ビットのクリア	クリアする	クリアしない	0
B1				
B0	FIFO メモリのクリア【注３】	クリアする	クリアしない	0

表３－13Ｂ【ＢＡＳＥ＋AH】出力ポートの構成

【注３】FIFO メモリ内容を全て破棄しフラグ出力も初期状態に戻す

３－15　ＦＩＦＯメモリ保持データ数読込み

```
ｍｅｍ_ｎｕｍｂｅｒ ＝ ０;
ｆｏｒ（ｌｏｏｐ ＝ ０；ｌｏｏｐ ＜ ４；ｌｏｏｐ++)
        ｍｅｍ_ｎｕｍｂｅｒ |=(ｉｎｐ(ＢＡＳＥ+０ｘ８)<< （８＊ｌｏｏｐ）);
```

本ボード内部では、常にＦＩＦＯメモリに格納されているデータ数を更新しています。更新単位はワードなので上記ポートから読み出した数値は、そのまま内部ＦＩＦＯに格納されているデータ数と一致します。また内部出力バッファには、次の読出しコマンドに供えてほとんど常に１バイトのデータが用意されていますが、この情報によって、上の式で求めた値を調整することはあまり意味がありません。
また、これらの機能もマイクロサイエンス製のボードでは使用できません。

３－16　メモリサイズ判定

```
ｍｅｍｅｍｓｉｚｅ＝ｉｎｐ（ＢＡＳＥ＋０ｘ９）;
```

このポートから読み出す値によりボード上のメモリサイズを判定する事ができます。現在対応している値は、０ｘ８０（１６M語）となっています。

３－17　ＡＤデータ読出し

```
ＡｄＤａｔａ＝ｉｎｐ（ＢＡＳＥ＋０ｘ０）;
ＡｄＤａｔａ｜＝ｉｎｐ（ＢＡＳＥ＋０ｘ０）＜＜８；　// ＡｄＤａｔａは読み出したAD変換データ
```

AD変換されたアナログデータはボード内のＦＩＦＯメモリに格納されており、アプリケーション側からはオフセット０からの連続した２バイトの読出しによってデータを引き取ります。仕組み的には、ＣＰＵのブロック転送命令を使用する事も不可能ではありませんが、Windows７以降のＯＳではハンドラー側で対応する必要があります。

３－18　割り込み制御（要因設定）

```
ｏｕｔｐ（ＢＡＳＥ＋０ｘ４、ＩｎｔＭａｓｋ）；　// 割り込み要因制御パターン：ＩｎｔＭａｓｋ
```

表３－１６【BASE＋０ｘ４】出力ポートの構成

ビット	各ビットの機能・意味	＝１の時	＝０の時	リセット時
B7	外部割込み信号の有効極性指定	↑（+）	↓（−）	0
B6	HalfFull 状態になった事による割り込み	許可	禁止	0
B5	NotEmpty 状態になった事による割り込み	許可	禁止	0
B4	未使用			
B3	１回サンプリングスキャン終了による割り込み	許可	禁止	0
B2	トリガ発生による割り込み	許可	禁止	0
B1	外部割込み信号による割り込み	許可	禁止	0
B0	連続サンプリングクロックによる割り込み	許可	禁止	0

B7：外部割込み信号が許可された場合の信号エッジ極性（↑↓）指定
B6：FIFO メモリ内の取り込みデータがハーフサイズを超えた状態になったタイミングによる割り込み制御
B5：FIFO メモリが空の状態から１データ取り込んだタイミングによる割り込み制御
B3：各回サンプリングスキャン終了タイミングによる割り込み制御
B2：トリガ発生による割り込み制御
B1：外部割込み信号による割り込み制御
B0：指定クロックの有効エッジによる割り込み制御

３－19　割り込み要求クリア

ｏｕｔｐ（ＢＡＳＥ＋０ｘ１、１）；　// 割り込み要求信号出力許可
ｏｕｔｐ（ＢＡＳＥ＋０ｘ１、０）；　// 割り込み要求信号クリア（出力禁止）
本ボードからの割り込み要求信号はソフトウェア上（デバイスドライバー内）でクリアする必要があります。PCI バスの割り込み信号がクリア操作まで割り込み要求状態を保持するレベル動作であるためです。
またこのポートはラッチポートですので、割り込み処理ルーチンの先頭で、ビット０を使用して一旦クリア操作を行うと、その状態は保持され次の割り込み要求は出力することができません。これを解消するには、割り込み処理を起動した割り込み要求がクリアされた後、再度ビット０をセットする操作が必要です。この一連の操作はデバイスドライバー内部で行われます。

ビット	各ビットの機能・意味	＝１の時	＝０の時	リセット時
B7～B1	未使用			
B0	PCI バスへの割り込み信号出力制御	出力許可	出力禁止	0

表３－１７【BASE＋２H】出力ポートの構成

３－20　汎用デジタル入出力

Ｄｉｎ＝ｉｎｐ（ＢＡＳＥ＋０ｘＡ）；　// 汎用１ビット TTL 入力（現在値）

ビット	各ビットの機能・意味	＝１の時	＝０の時
B7～B1	未使用		
B0	Ｉ０：汎用デジタル入力ビット０	Ｈｉｇｈ（開放）	Ｌｏｗ（０Ｖレベル）

表３－１８Ａ【ＢＡＳＥ＋AH】入力ポートの構成

ｏｕｔｐ（ＢＡＳＥ＋０ｘＡ、Ｄｏｕｔ）；　// 汎用１ビットラッチ出力

ビット	各ビットの機能・意味	＝１の時	＝０の時	リセット時
B7～B1	未使用			
B0	Ｑ０：汎用デジタル出力ビット０	Ｈｉｇｈ	Ｌｏｗ	0

表３－１８Ｂ【ＢＡＳＥ＋AH】出力ポートの構成

表中の値はＳ－ＰＯＬジャンパーが“P”側に設定されている場合です。
【注１】出力素子はＳＮ７５４５２ＢＰを使用しています。（耐圧＝３０Ｖ、出力電流３００ｍAmax
デバイスそのものは出力電流３００ｍAまで使用する事が可能ですが、その時の出力電圧がＴＴＬローレベルの許容範囲を超えてしまうため１００ｍAまでとします。）
【注２】電源投入またはハードウェアリセット直後の汎用デジタル出力の状態は“０”ですが、ソフト的な制御部リセット操作ではその状態は変わりません。
【注３】汎用デジタル出力の論理はボード上のジャンパーＳ－ＰＯＬによって任意に設定することができます。また出力回路については、ボード上のジャンパーＳ－ＰＵＰによってオープンコレクタ（O 側）または１KΩのプルアップ（U側：標準）を選択できます。

３－21　サンプルホールド回路制御（オプション）

本ボードでは、外付のサンプルホールド回路を利用して 16 チャンネルまでの同時サンプリング動作を行わせる事が可能です。その際、サンプルホールド回路側からの要請によりサンプルホールド開始から先頭チャンネルのサンプリングまでの間に凡そ 1uS 程度のアクイジションタイムが必要になります。この待ち時間を挿入する事で先頭チャンネルのサンプリング誤差が悪化する事を避ける事ができます。しかしながらその結果最高サンプリング速度について若干の速度低下が発生しますので､本機能によりこの待ち時間有無を切り替える事ができます。

実際のレジスタマッピングを図３－19 に示します。

ビット	各ビットの機能・意味	＝１の時	＝０の時	リセット時
B7～B1	未使用			
B0	待ち時間追加の有無	待ち時間追加	待ち時間なし	0

図３－19【ＢＡＳＥ＋ＣＨ】出力ポートの構成

３－22　マスタースレーブ接続説明（予約事項）

本ボードを複数枚使用して同期運転することができます。この場合、トリガを検出するボードをマスター、他のボードをスレーブとします。スレーブはマスターからのクロックを受けて同期運転を行いますが、サンプリングタイミングに最大 125nS の遅れが生じます。

マスタースレーブ間の接続及び操作については次の通りとなります。

ボード上の設定

マスタースレーブの棲み分けは、ボード上にあるロータリースイッチ（SW-BN）によって行う。マスターボードでは、スイッチの設定は“０”とし、スレーブボードでは非“０”とする。ハンドラーからはスイッチの値が“０”であるマスターが第一番に、以下“１”であるスレーブ、“２”であるスレーブという順番で認識され扱うアナログデータについてもこの順番によって処理される。

ボード間の接続

マスターボードのクロック出力をスレーブボードのクロック入力に接続する。マスターボードについては、全てのトリガ方式を使用することができる。

ソフトウェア設定

スレーブボードは全てクロック源を外部立下りとし、クロック源分周比も１／１（非分周）と設定、トリガーソースについては、ソフトウェアトリガのみとしアナログトリガ、外部トリガは禁止しておく。

連続サンプリング動作開始は、最初に（ソフトウェアトリガで）スレーブボードに対して行い、最後にマスターボードを任意のトリガ条件で起動する。

【注】これらソフトウェア設定の記述は汎用ハンドラーを用いてアプリケーションを組上げる際の指標として記載したものです。専用ハンドラーでは、関数セットの形態上現状未実装となっています。

※　関数セットの引数にボード番号が付与されていないため、現状未実装としています。

第４章 ソフトウェア

４－０　ハードウェアとソフトウェアの連携

前章までで解説したハードウェアと、ユーザーアプリケーションとを接続する役目を果たしているのが、本章で説明するデバイスドライバーと呼ばれるソフトウェアになります。その動作はかなり複雑ですが、視覚的に簡略化した相互の関係を図４－０に示します。

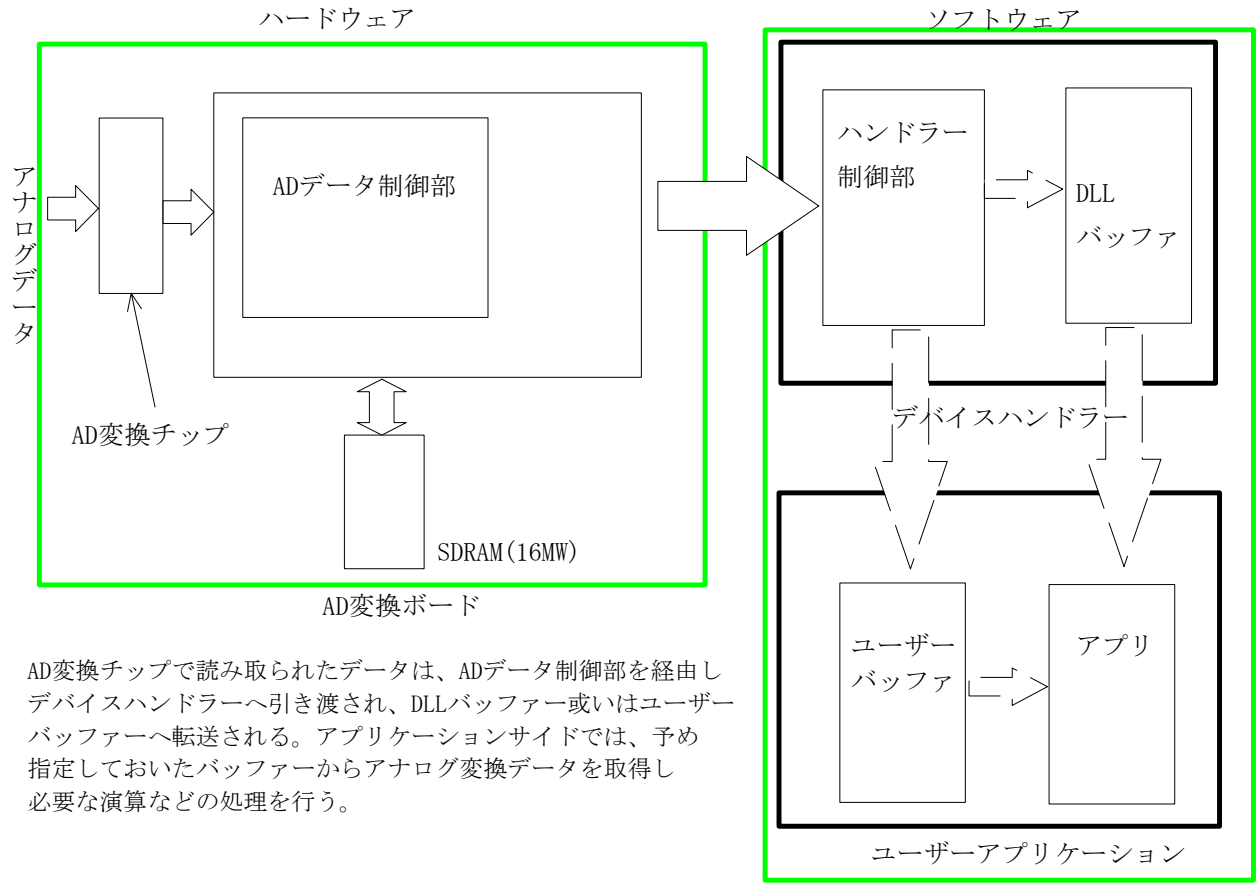


図４－０　ハードウェアｖｓユーザーアプリケーション相関図

AD 変換ボード内の全てのリソースは、ソフトウェアサイドから見ると入出力レジスタにマッピングされています。MSDOS の時代であれば、ユーザーアプリケーションサイドから直接入出力命令を発行する事により、ハードウェアの制御ができましたが、現在ではそのような操作はセキュリティ上の問題から禁止されており、これらレジスタの操作は、デバイスハンドラーというシステムサイドのソフトウェアを介在させて行うことになっています。

本章の以下の部分では、このデバイスドライバーを使用したハードウェア制御の方法を、順序だてて説明してゆく事になります。

尚、本デバイスドライバーはマイクロソフト製ビジュアルスタジオ 2008 を使用して製作されているため ビジュアルスタジオ 2008 用ランタイムライブラリが同時にインストールされます。

４－１　本製品用ソフトウェアの種類

本製品用のソフトウェアは、表４－１に示す３種類から構成されています。即ち、WindowsXP(32Bit)対応のデバイスドライバー、Windows7(32Bit)対応のデバイスドライバー及びWindows7(64Bit)対応のデバイスドライバーの３種類です。また、夫々のＯＳ用に専用ハンドラー並びに汎用ＩＯハンドラーが用意されています。

表４－１　ドライバー構成表

ＯＳ及びビット幅		Windows7 32Bit	Windows7 64Bit	WindowsXP 32Bit
パッケージ名称		Redist412_x86.zip	Redist412_x64.zip	Redist412_xp.zip
専用ドライバー	ドライバー(DLL/Lib)	FCAD412PCI_X86	FCAD412PCI_X64	FCAD412PCI_X86
	関数宣言	FCAD412PCI.H	FCAD412PCI.H	FCAD412PCI.H
汎用ドライバー	ドライバー(DLL/Lib)	PORT7_X86	PORT7_X64	PORT7_X86
	関数宣言	PORT7.H	PORT7.H	PORT7.H

WindowsXP(32Bit)対応のドライバーは、アプリケーション移植時の利便性を考慮し用意されているものです。このドライバーを経由する事により、ドライバー更新→ＯＳ変更というステップを２段階に分割して実行できるように考慮されています。ドライバーそのものはWindows7(32Bit)版と同じものになっていますが、インストール方法の違いによりパッケージの構成がWindowsXP対応の形になっています。
またドライバー内の関数セット名称の内、同一機能のものについてはマイクロサイエンスのドライバーと同一とし移植時の利便性を考慮しています。

４－２　ドライバーのインストール

４－２－１　WindowsXP ドライバーインストール及びアンインストール

WindowsXP対応ドライバーをダウンロードし、その内容を全て解凍します。解凍場所はＣドライブのルートを指定して解凍します。解凍結果は以下のようになります。

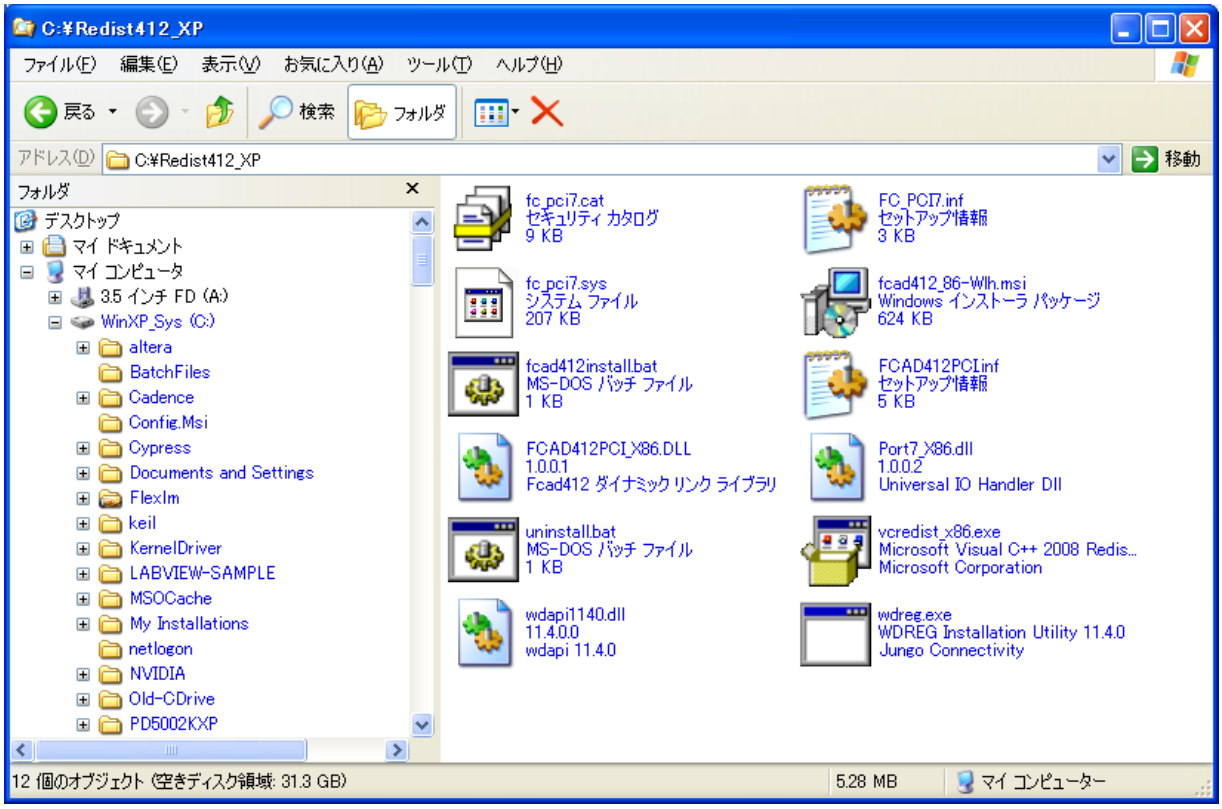


図４－２－１Ａ

デバイスインストールを開始する前に、旧ドライバーがインストールされていた場合、これを削除しなければなりません。そのためには、デバイスマネージャを開き、マイクロサイエンスドライバーが登録されているかど

うかを確認します。因みに、下の図は、参考例としてＨＤＡ－７７０ＰＣＩがインストールされている状態で説明しています。

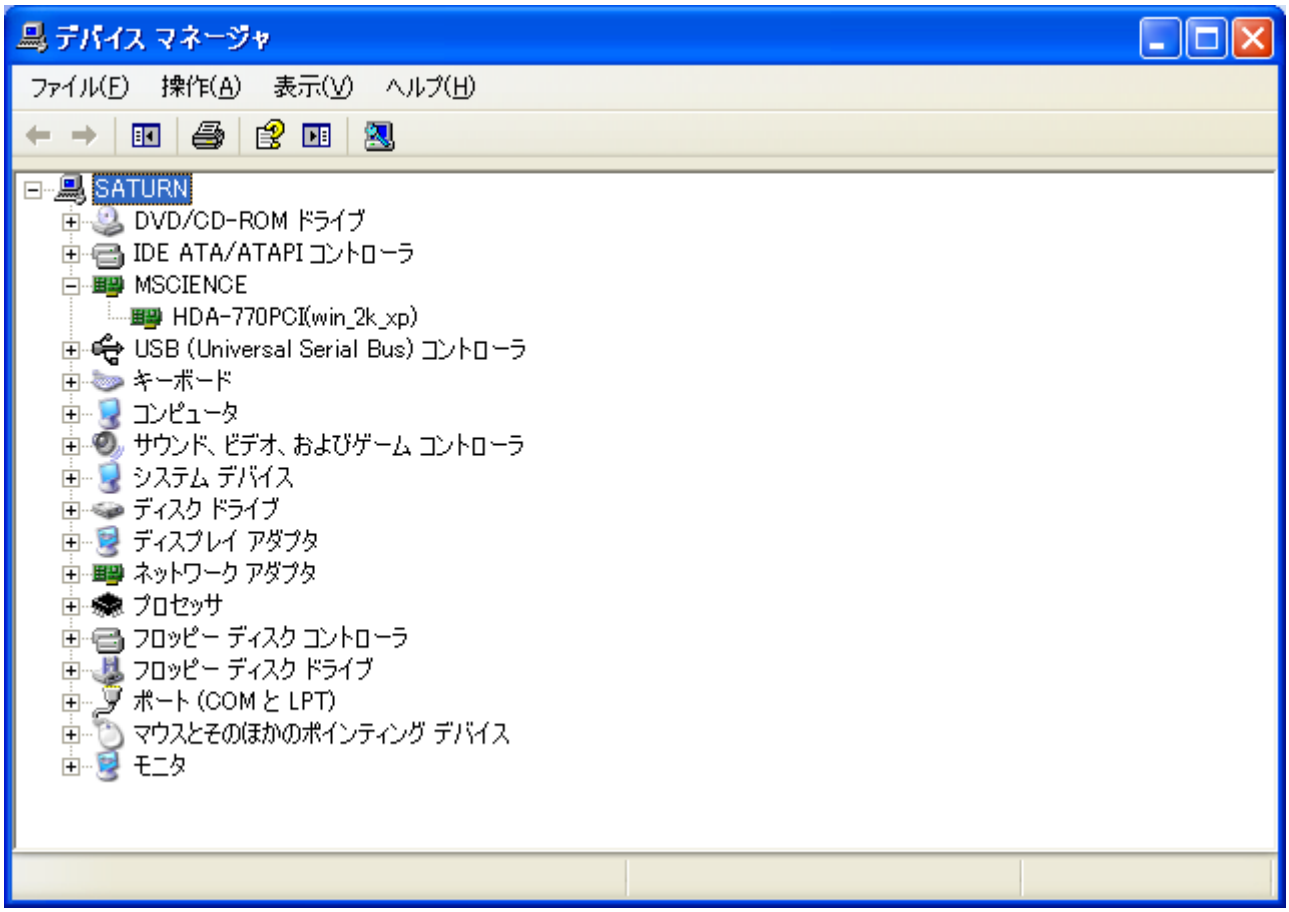


図４－２－１Ｂ

上図に示すようにMSCIENCEというエントリーが存在すれば、旧ドライバーがインストールされています。
ドライバー削除は、

1　まずWindows¥Infフォルダの中からOem*.Infファイルを検索します。検索条件としては、文字列として“MSCIENCE”が含まれているものを条件とします。検索されたファイルをテキストエディターでマイクロサイエンスのものである事を確認し、Oemxx.Inf（xxは上記の条件に合致した番号です）とOemxx.Pnfを削除します。
2　次にWindows¥System32フォルダの中から、DMS_PCI.SysとDMS_PCI.Infを探し同じように削除します。
3　最後に、図４－２－１Ｂで示すマイクロサイエンス製ドライバー（例ではHAD-770PCIです）を右クリックし削除します。

ここまできた段階で一旦パソコンの電源を落とし、ボードを取り外します。
再度パソコンの電源を投入し、図４－２－１Ａで示されるフォルダの中から“Fcad412PciInstall.Bat”ファイルを右クリックし“開く”（実行する）を選択します。すると、まず図４－２－１Ｃで示されるウィンドーが開きます。（次ページ）

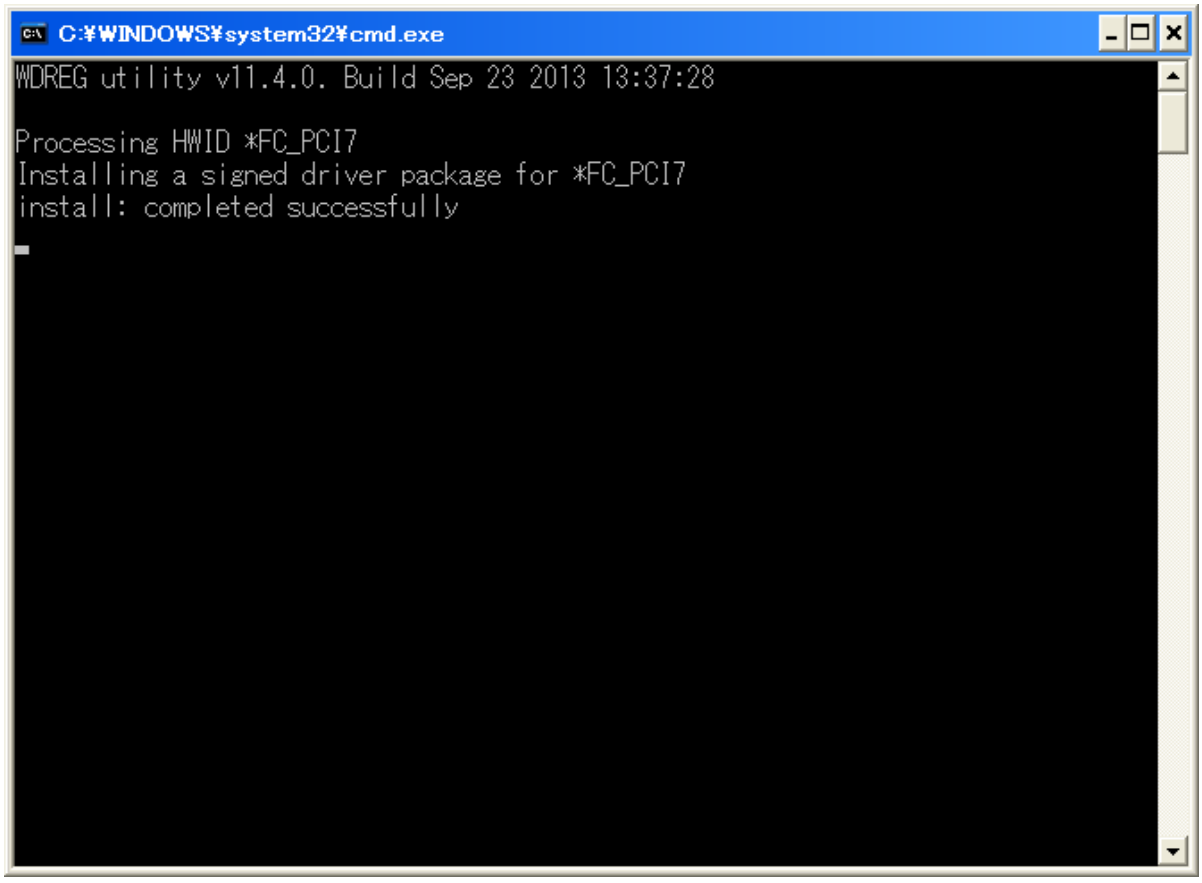
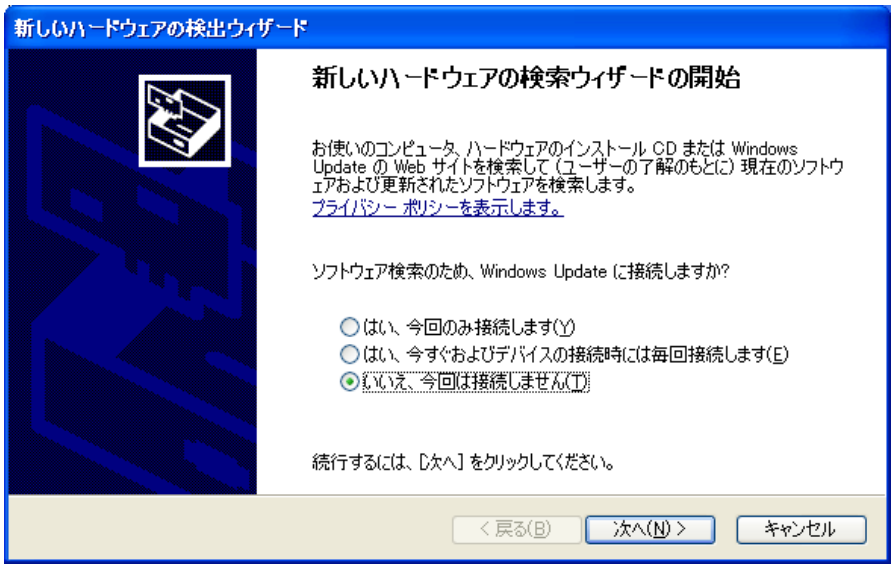
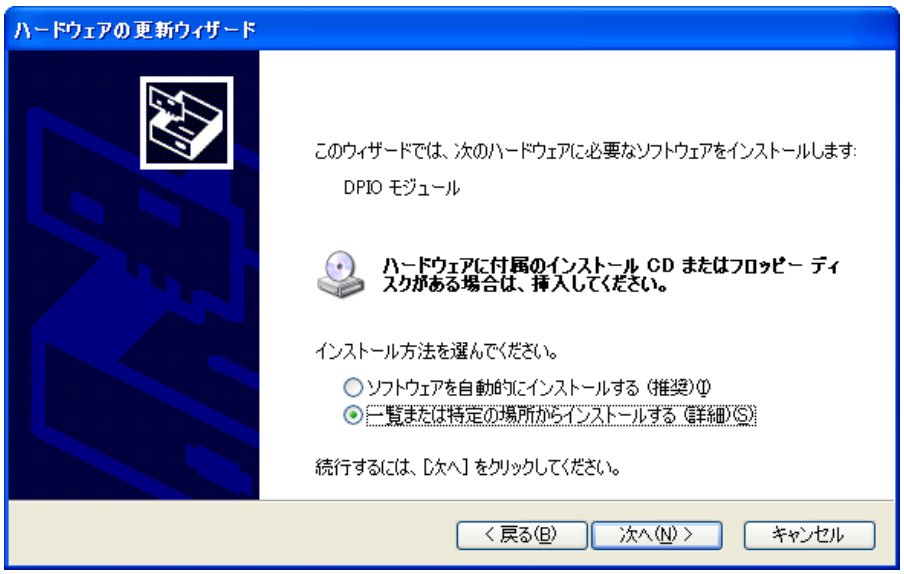


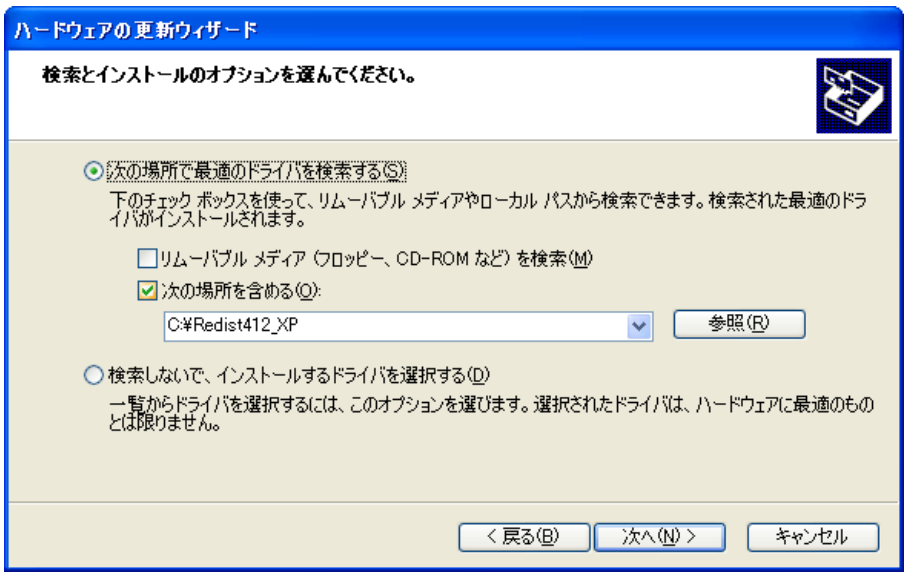
図４－２－１C

そして暫くすると、インストール作業が終了します。そこで再度パソコンの電源を遮断し FCAD412PCI ボードを再度挿入し電源を投入します。システムが立ち上がるとハードウェアウィザードが立ち上がりますから、



WindowsUpdate への接続をせず、“次へ“ボタンをクリックします。そして次の画面（次頁）で、次のハードウェアとして” FCAD412PCI” が表示された場合は、インストール方法として“ソフトウェアを自動的にインストールする”を選択し“次へ“ボタンをクリックします。すると必要な Dll ファイル群がシステムフォルダにコピーされインストール作業は終了します。また次頁のようにハードウェアとして”DPIO モジュール”が表示された場合は”一覧または特定の場所からインストールする“を選択し”次へ“ボタン押下後の画面でインストール先として“C:¥Redist412_XP”を選択しインストールを行います。





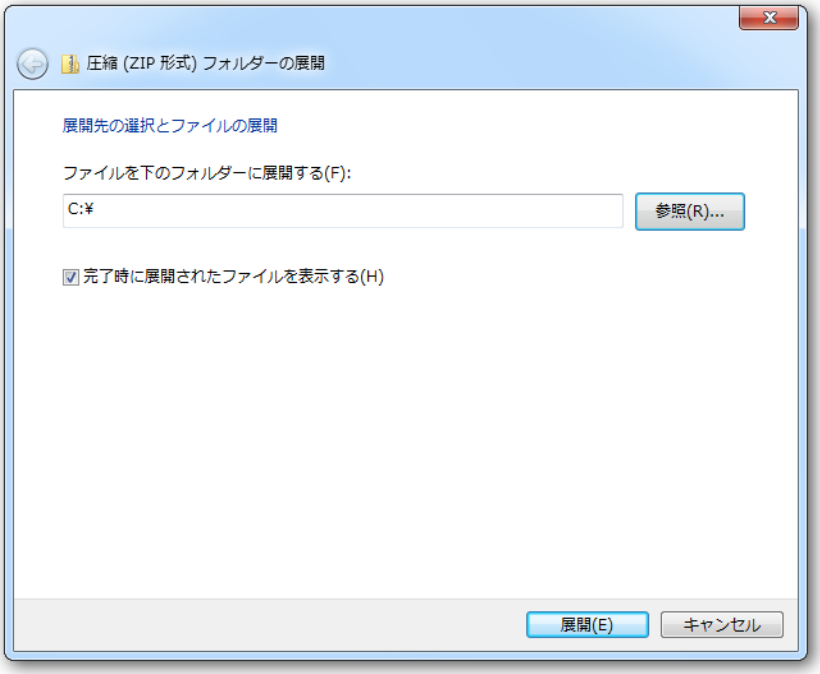
また、ドライバーを削除する場合は、図４－２－１Ａで“Uninstall.Bat”ファイルを実行することで自動的にドライバーが削除されます。

４－２－２　Windows７（X86：32 ビット）へのインストール

本製品用のソフトウェアは弊社ホームページ内ダウンロードコーナーにアップロードされています。その中から、実際に御使用になるＯＳとマッチしたドライバーセットをダウンロードして頂き本ボードを取り付ける前にデバイスドライバーをインストールしていただきます。

実際のインストール方法は以下の通りです。

・　本ボードをパソコンにセットアップする前に、弊社ホームページよりご使用のＯＳ用デバイスドライバーキットをダウンロードして頂きパソコンの中に解凍します。解凍場所は、Ｃドライブのルートとします。（本ドライバーキットには、デバイスハンドラーが含まれています。）



前ページの“展開”ボタンをクリックするとドライバーセットがＣドライブ内に展開されます。内容は図４－２－２Ａを参照してください。

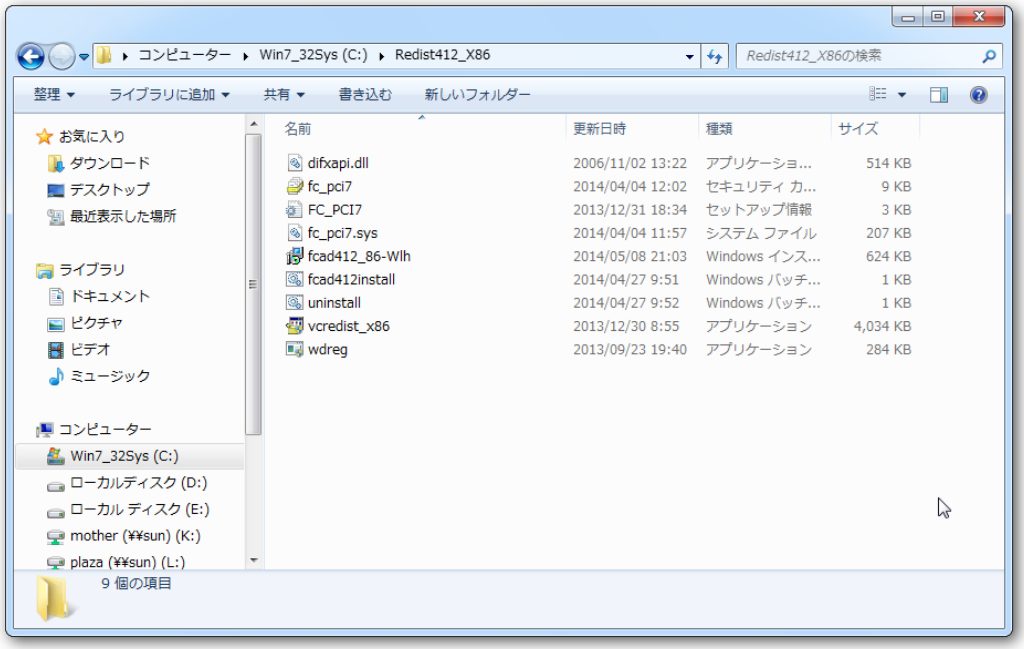
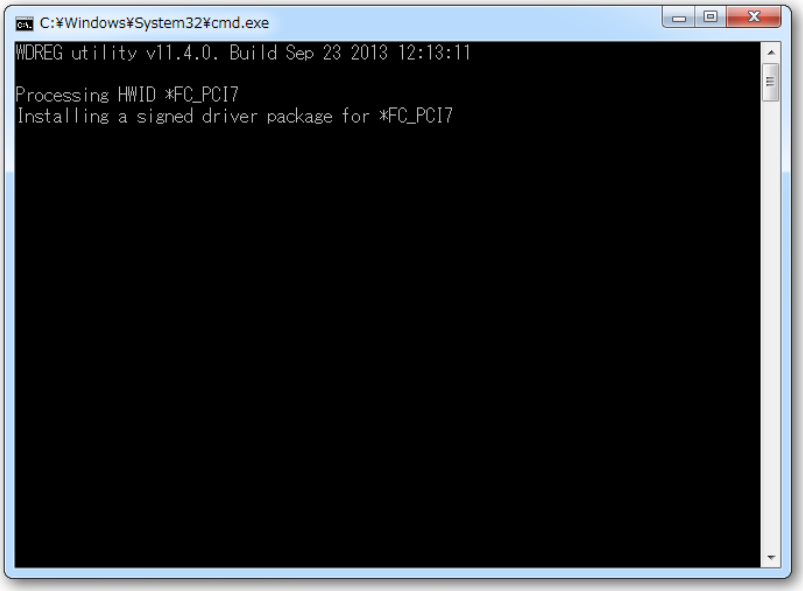
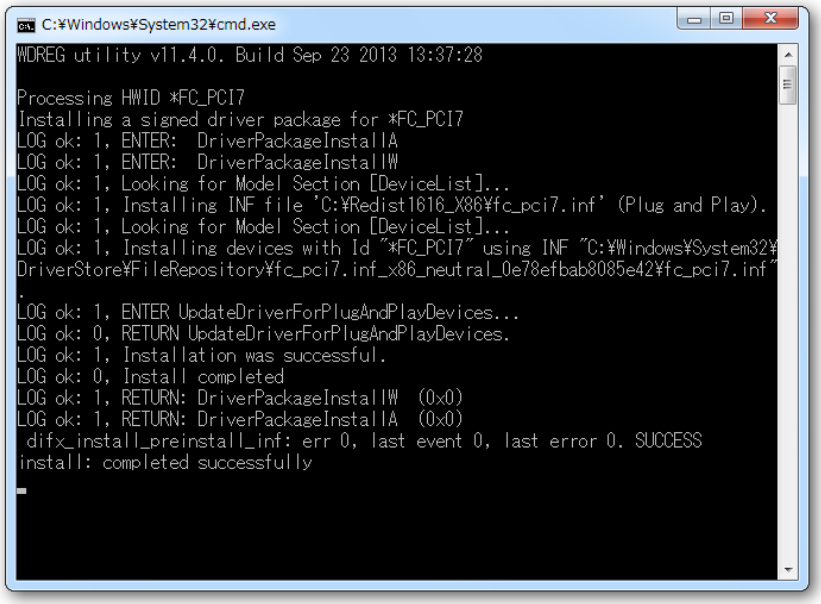


図４－２－２Ａ

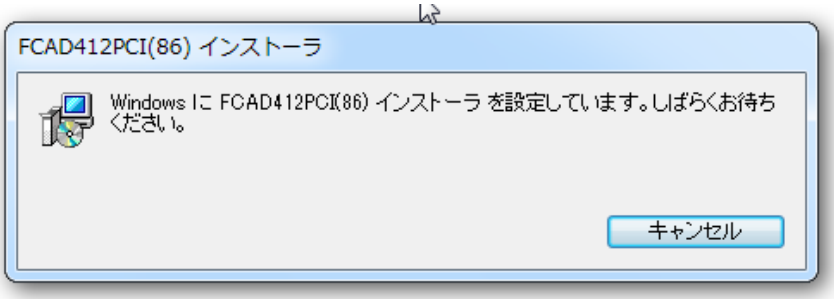
この中で“Fcad412Install.Bat”がデバイスドライバーインストーラーです。***また、この時点で、本ボードは、まだパソコンには取り付けないで***下さい。この順序が守られない場合、正常なインストールは保証されません。このデバイスドライバーインストーラーを***管理者として***実行（右クリックメニューから選択）すると次のウィンドーが開きます。（ただ実行とすると、ウィンドーが瞬間的に終了しデバイスドライバーのインストールはできません）最初に次のウィンドーが開きます。



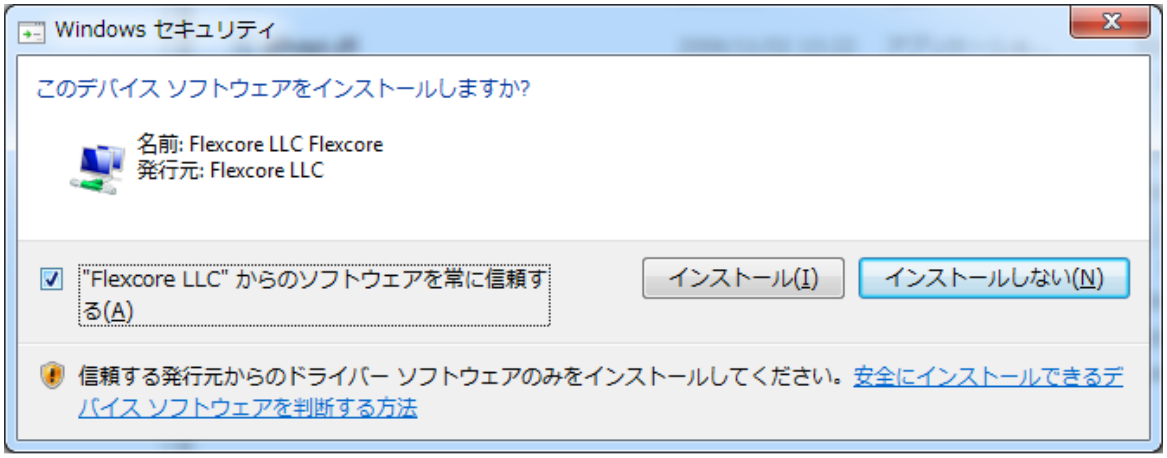
次いで次ページに示すウィンドーが開きます。



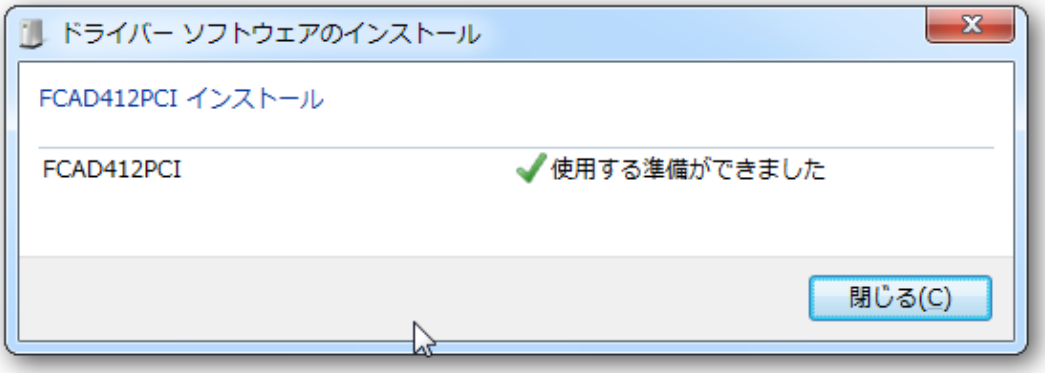
更に暫く待っていると、次のウィンドーが開きます。



更に待っていると次のウィンドーが開きます。



ここでは、“Ｆｌｅｘｃｏｒｅ　ＬＬＣ”からのソフトウェアを常に信頼するという項目にチェックを入れ“インストール”ボタンをクリックします。そして暫く待っているとインストール作業が終了します。
この後、一旦パソコンの電源を落とし、パソコン内部のＰＣＩスロット或いはＰＣＩ拡張ユニットに本ボードを挿入し再度パソコンの電源を投入します。ＯＳが立ち上がるとハードウェアの検出が行われ、少し待っていると画面下部のタスクバーの所に、デバイスドライバーが正常にインストールされたというメッセージが表示されます。



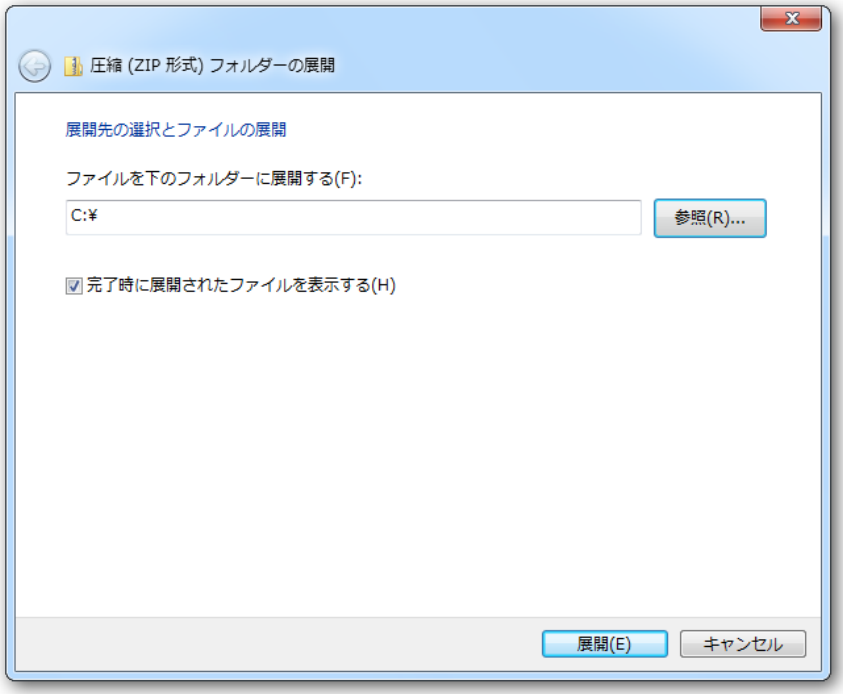
これで本ユニットのインストールは完了しました。

４－２－３　Windows７（X64：64 ビット）へのインストール

本製品用のソフトウェアは弊社ホームページ内ダウンロードコーナーにアップロードされています。その中から、実際に御使用になるＯＳとマッチしたドライバーセットをダウンロードして頂き本ボードを取り付ける前にデバイスドライバーをインストールしていただきます。

実際のインストール方法は以下の通りです。

・　本ボードをパソコンにセットアップする前に、弊社ホームページよりご使用のＯＳ用デバイスドライバーキットをダウンロードして頂きパソコンの中に解凍します。解凍場所は、Ｃドライブのルートとします。（本ドライバーキットには、デバイスハンドラーが含まれています。）



ここで“展開”ボタンをクリックするとドライバーセットがＣドライブ内に展開されます。内容は図４－２－３Ａを参照してください。

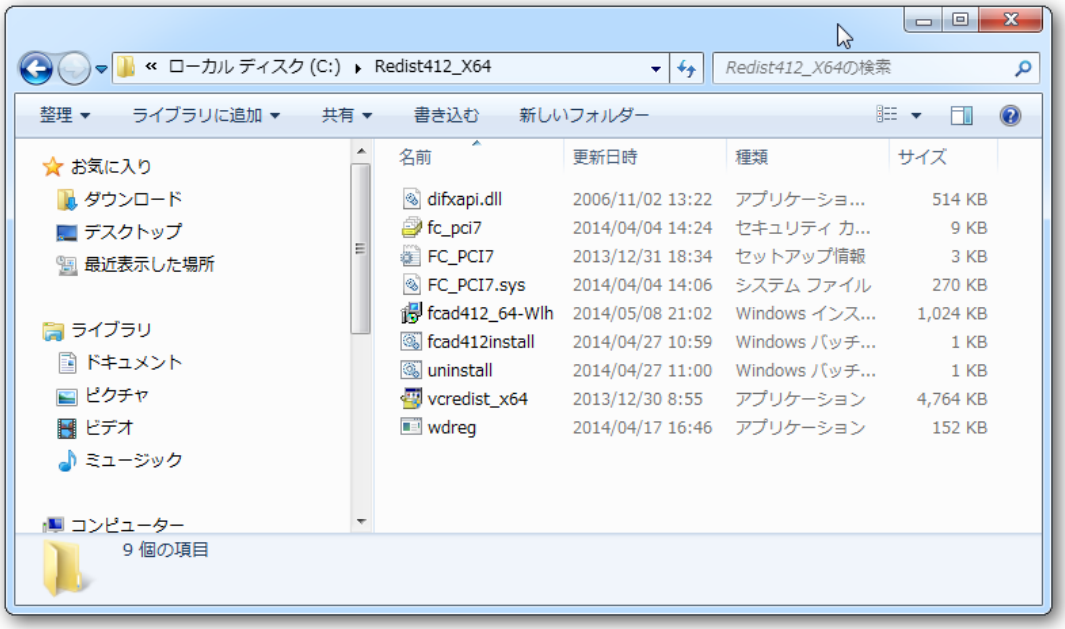
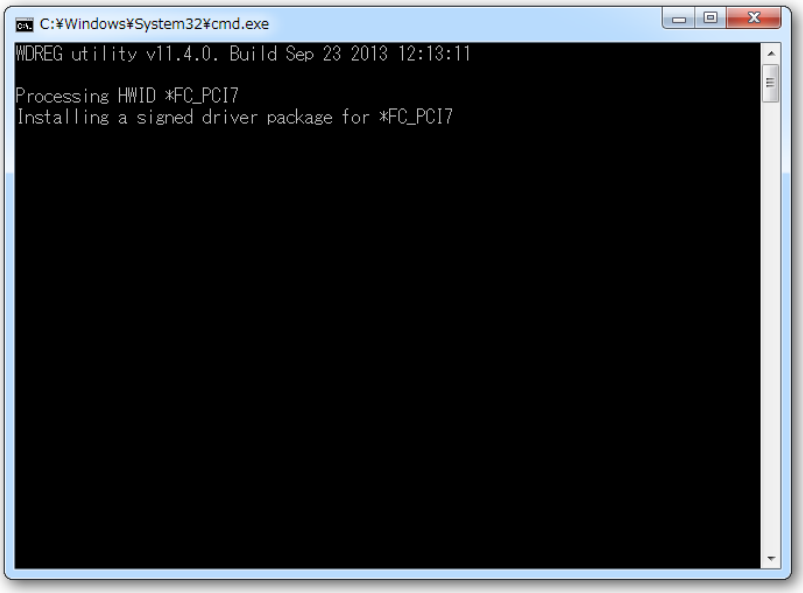


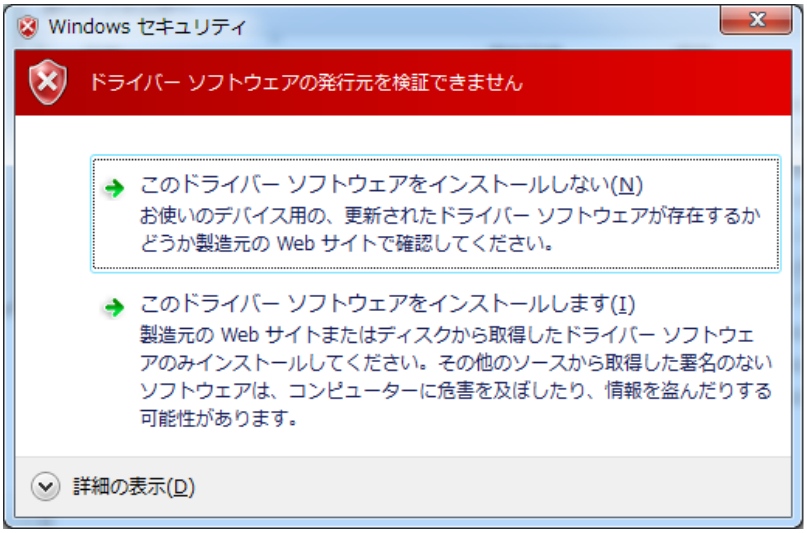
図４－２－３Ａ

この中で“Fcad412Install.Bat”がデバイスドライバーインストーラーです。***また、この時点で、本ボードは、まだパソコンには取り付けないで***下さい。この順序が守られない場合、正常なインストールは保証されません。このデバイスドライバーインストーラーを***管理者として***実行（右クリックメニューから選択）すると次のウィンドーが開きます。（ただ実行とすると、ウィンドーが瞬間的に終了しデバイスドライバーのインストールはできません）

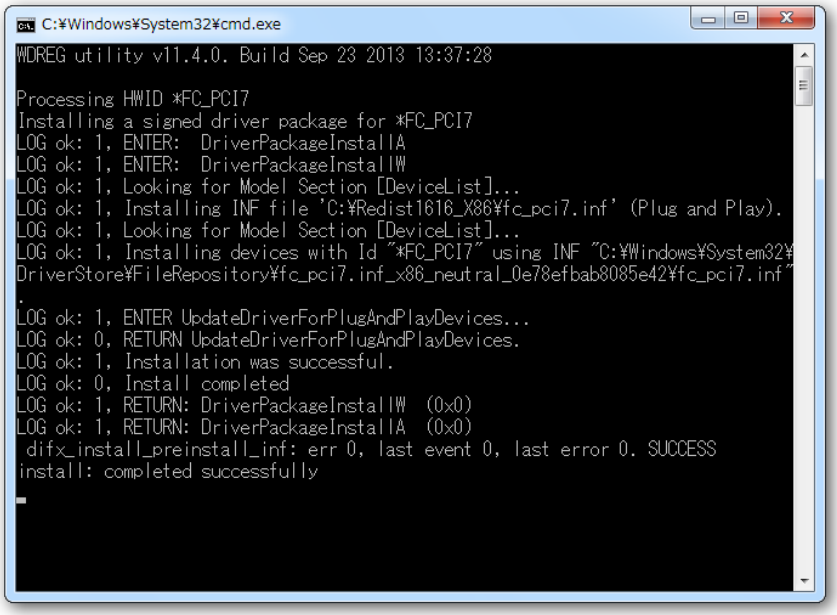
最初に次のウィンドーが開きます。



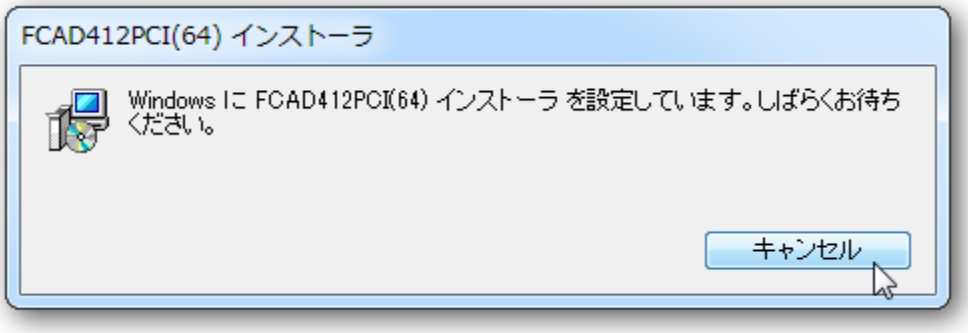
次いで次ページに示すウィンドーが開きます。



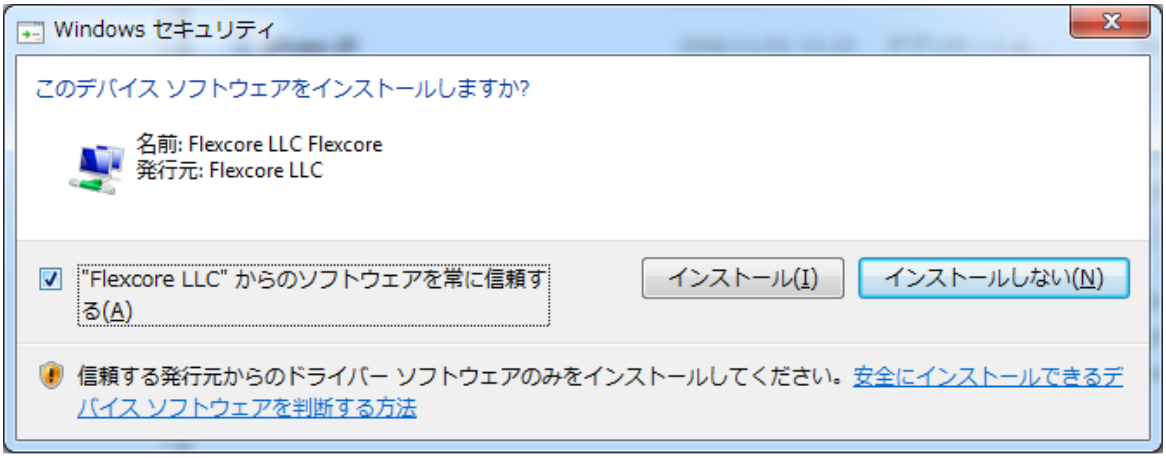
ここでは、“このドライバーソフトウェアをインストールします”側のボタンをクリックして下さい。すると次のウィンドーが表示されます。（次ページ）



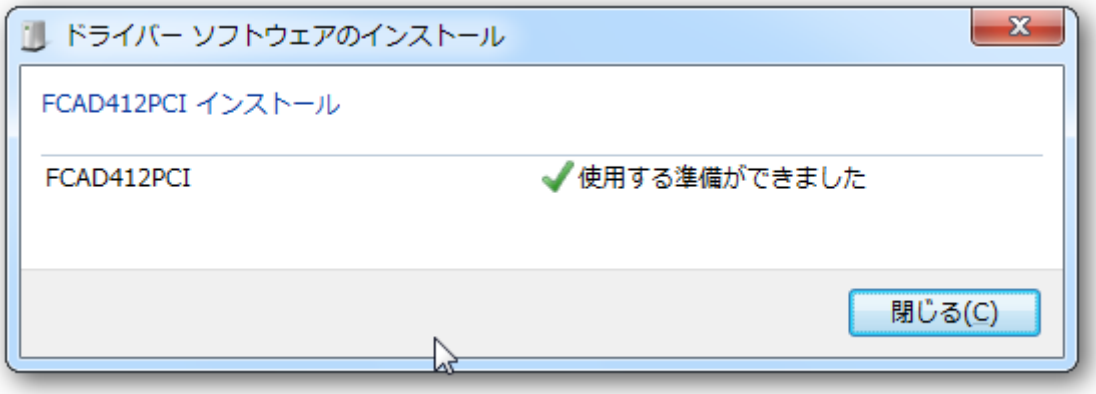
更に暫く待っていると、次のウィンドーが開きます。



更に待っていると次のウィンドーが開きます。



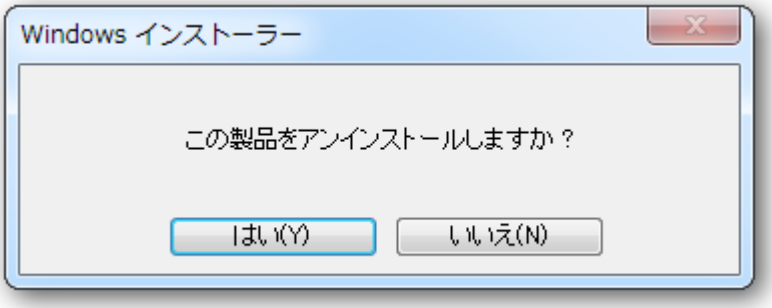
ここでは、“Ｆｌｅｘｃｏｒｅ　ＬＬＣ”からのソフトウェアを常に信頼するという項目にチェックを入れ“インストール”ボタンをクリックします。そして暫く待っているとインストール作業が終了します。
この後、一旦パソコンの電源を落とし、パソコン内部のＰＣＩスロット或いはＰＣＩ拡張ユニットに本ボードを挿入し再度パソコンの電源を投入します。ＯＳが立ち上がるとハードウェアの検出が行われ、少し待っていると画面下部のタスクバーの所に、デバイスドライバーが正常にインストールされたというメッセージが表示されます。



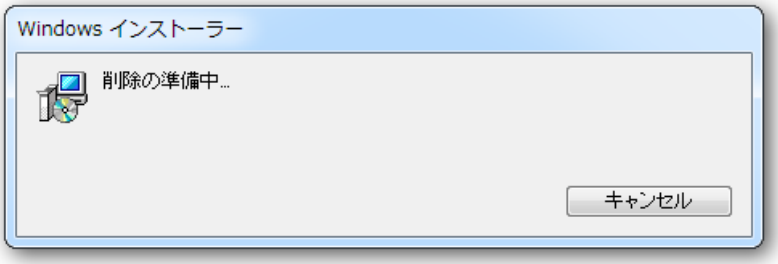
これで本ユニットのインストールは完了しました。

４－３　Windows７ドライバーのアンインストール

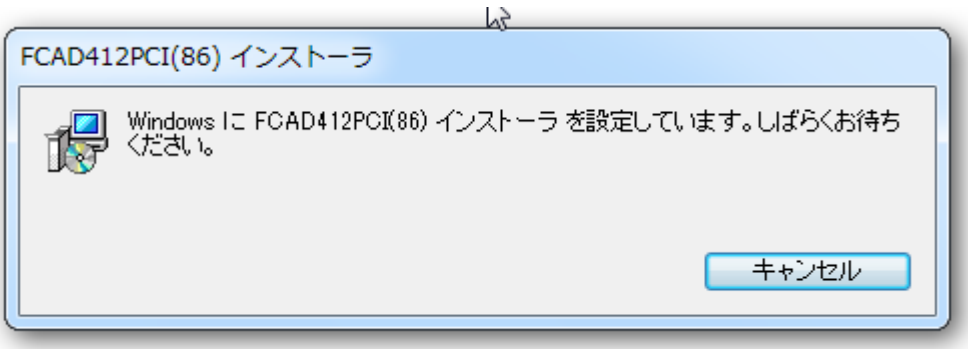
この方式でインストールしたドライバーはドライバー更新時などにドライバーをアンインストールする際にも簡単に行う事ができます。まず、４－２－１項で説明したデバイスドライバー解凍フォルダへ移動し、この中にある“Uninstall.Bat”ファイルを実行します。（ここでも*管理者として*実行する必要があります。）



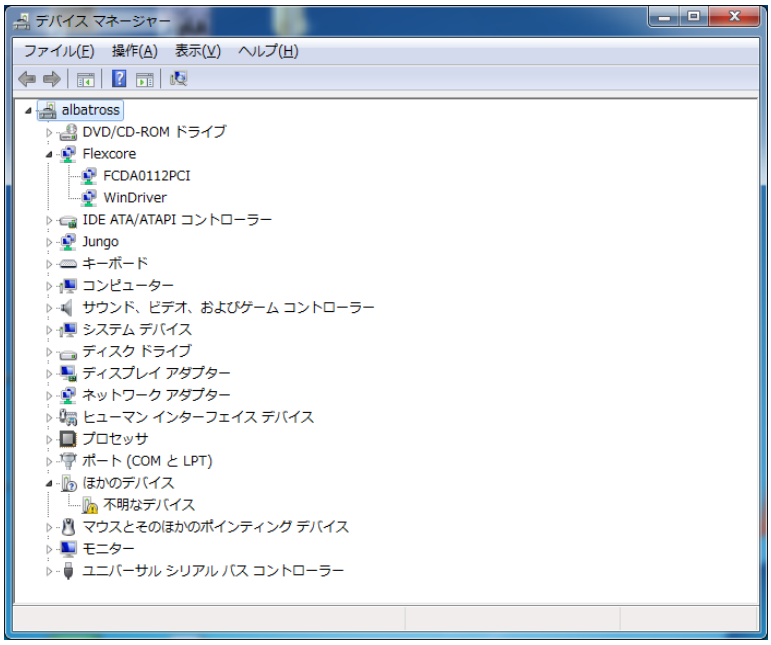
というメッセージが現れますので“はい”をクリックします。すると



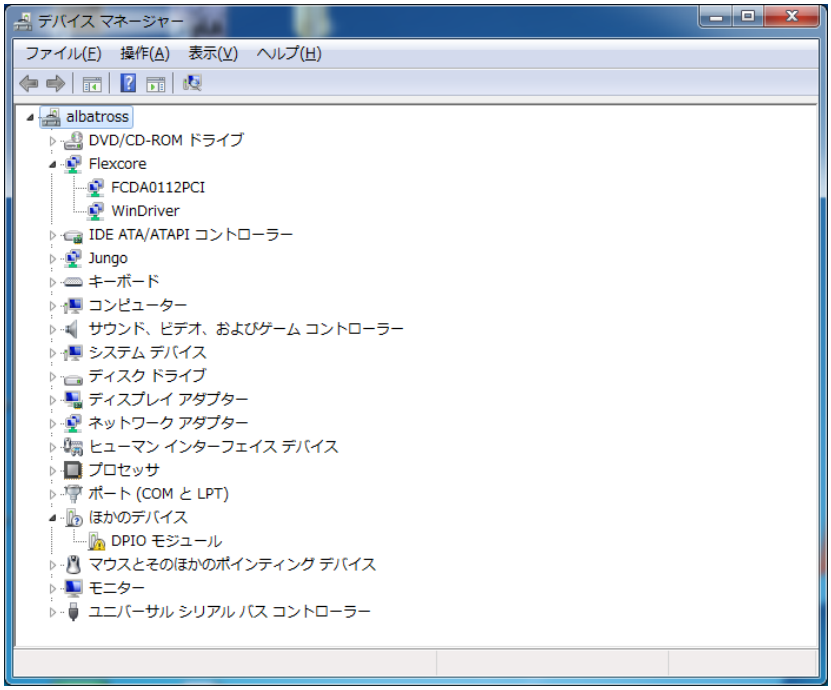
というウィンドーに変わり、更に待っていると



というウィンドーに変わり、暫く待っているとデバイスドライバーが削除されます。
この段階で、スタートー＞コントロールパネルと進み、デバイスマネージャを開くと、下図に示すように不明なデバイスが表示されています。（同図下から４行目）
他方、上から４行目にはアンインストールされていないボード情報も存在しています。



そこで、この不明なデバイスを右クリックし、表示されるドロップダウンリストから“デバイスの削除”を選び実行します。その結果、次ページに示すように不明なデバイスが、ドライバー未インストール時と同様、ほかのデバイスの中にＤＰＩＯモジュールが存在する状態に変化します。（この表示を得るためには、トップエントリ：同図では albatross と表示されているエントリーを右クリックし“ハードウェア変更のスキャン”を実行する必要があります）
この状態に到達すると、パソコンの電源を遮断し、本ボードを取り外す事ができます。
　特に、バージョンの異なるドライバー（例えばWindows8 対応のドライバー）や、他メーカーのドライバーに切り替える時には、下の図で示されるように、本ボードが DPIO モジュールという状態になっている必要があります。



この状態で、電源遮断後、本ボードをシステムから取り除くと、４－２項　ドライバーインストール直前の状態に復帰している事になります。

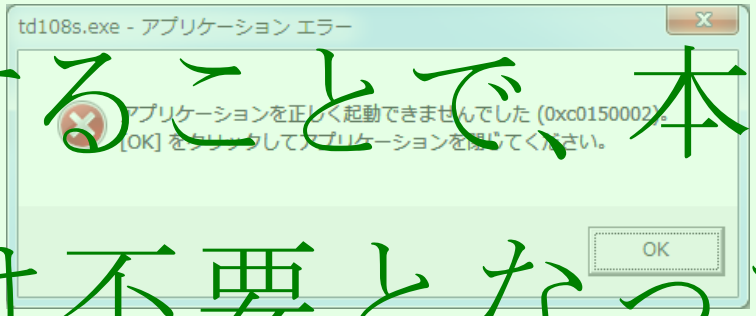
４－４　サンプルプログラムについて

本製品のサンプルプログラムは、大きく２種類から構成されています。第一のプログラムは、汎用入出力関数を利用したボードの動作確認プログラムであり、(内部レジスタ動作確認及び簡単な AD データ取り込み処理)第二のプログラムは専用ハンドラーを使用した C+による自動サンプリング（Dll バッファ使用)、マニュアルサンプリング（ユーザーバッファ使用）及びループ処理プログラムの３構成から構成されています。また、アナログ回路設定変更などの実行例については、基本的に自動サンプリングサンプルの中で記述しています。また、X86（32 ビット)及び X64(64 ビット)の実行ファイルについてはそれぞれのサンプルプログラム内でディレクトリ構造によって住み分けを行っています。詳細については、弊社ホームページよりダウンロードし内容の確認をお願いします。

４－５　サンプルプログラム、動作確認プログラムについて（補足）

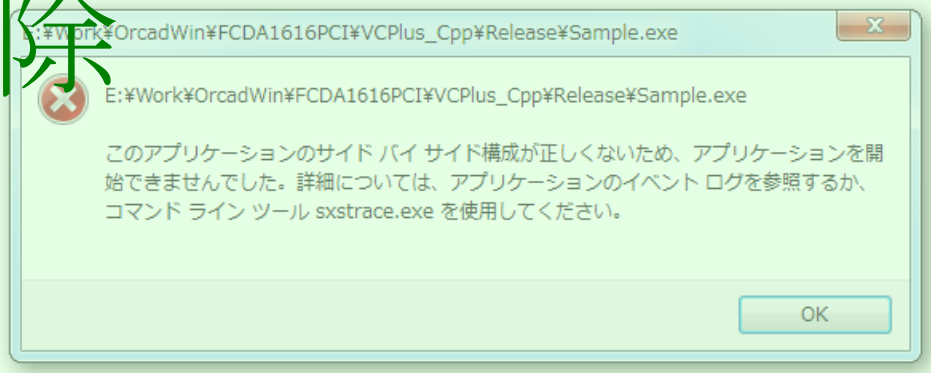
動作確認プログラムをWindows7の32ビット環境で動作させると、パソコン側の設定によってはエラーメッセージが表示されプログラムが動作しない事があります。（実際のメッセージは次ページに示します。）これはATL関連のアップデートと関係があるようで、弊社の環境ではWindows7（32ビット）をインストールしているがVS2008がセットアップされていない場合表示されました。このエラーはVS2008のランタイムをインストールしても回避できず、VS2008をインストールすると（SP1パッチを適用する前であっても）回避する事ができました。同じWindows7であっても64ビット版ではVS2008インストールをしなくてもこのエラーは現れないので、恐らくＯＳ自身の構造と関係があるものと思われますが、詳細は不明です。

本現象は、コンパイルオプションの設定ミスにより本来は静的ランタイムが使用される設定になっていたためにVS2008をインストールする事で正常な状態に戻るという事が判明したため、オプションを設定しなおす事で対策をとる事ができました。



動作確認プログラム実行不可能を示すメッセージ

また、サンプルプログラムを実際に動作させると、以下のようなメッセージが出て動作しない事があります。これは、Windows７若しくはは WindowsXPにおいてＣ＋ランタイムが標準でインストールされていないためです。



Windows7の場合

WindowsXPの場合

弊社で作成した動作確認プログラムはＭＳＶＣ＋２００８でコンパイルされておりますので、専用のランタイムをインストールする事により、この状況を回避する事ができます。必要となるランタイムは、ドライバーパッケージの中に同梱しておりますので、こちらを実行することでランタイムのインストールができます。ファイル名称は、ｖｃｒｅｄｉｓｔ_ｘ８６.ｅｘｅ（Ｘ８６環境）又はｖｃｒｅｄｉｓｔ_ｘ６４.ｅｘｅ（Ｘ６４環境）になります。

若しくは、（まだＣ＋環境をインストールされていない場合）当該パソコンにＭＳＶＣ＋２００８をインストールする事でも対処する事ができます。

VS2008ランタイムライブラリを自動インストールすることで、本注意事項は不要となったため削除

４－６　関数仕様、エラーコード

連番	関数名	機能・動作	パラメータ等
【１】	AD682_Open_ADSys	ボード、本ハンドラーの初期化	使用ボード数
【２】	AD682_Set_SampCh	サンプリング実行チャンネル数設定	実行チャンネル数
【３】	AD682_Set_SampMode	サンプリングモードの設定	バッファ領域
【４】	AD682_Set_Trigger	トリガ関連の設定	トリガ源、極性
【５】	AD682_Set_RangeTrigger	アナログレンジトリガ関連の設定	同上
【６】	AD682_Set_Exclk	オプション外部クロック値の設定	外部クロック周波数
【７】	AD682_Set_Clock	サンプリングクロック値の設定	
【８】	AD682_Start_Samp	サンプリングスタート	
【９】	AD682_Get_Status	ボードステータス取得	ボードステータス
【10】	AD682_Close_ADSys	本ハンドラーの終了	
【11】	AD682_Read_DLLData	DLL 管理バッファからのデータ読出し	波形データ入力
【12】	AD682_Read_DirectFifo	ユーザーメモリへの直接データ転送	波形データ入力
【13】	AD682_Get_OneScan	マニュアルサンプリング	ＡＤサンプリングデータ
【14】	AD682_Out_Aux	汎用デジタル出力	更新データ
【15】	AD682_Inp_Aux	汎用デジタル入力	
【16】	AD682_Set_SampLoop	リングバッファ指定有無	
【17】	AD682_Set_Inpmode	アナログ入力モード設定	
【18】	AD682_Stop_Samp	サンプリングの強制停止	
【19】	AD682_Read_RestData	残り AD データの読込み	
【20】	AD682_Clear_Flags	各フラグのクリア	バイナリ／2 の補数等
【21】	AD682_Set_Shc	外付同時サンプルホールド制御	
【22】	AD682_Get_Libver	本ハンドラーのバージョン情報取得	
以下に記述した４種類の関数はＦＣＡＤ４１２独自の機能を使用するための関数セットである			
【23】	AD412_Get_HalfSize	ハーフフルサイズ取得	
【24】	AD412_Get_UsedSize	メモリ使用データ数の取得	
【25】	AD412_Set_DataCode	AD データコード、入力モード等の設定	
【26】	AD412_Get_DataCode	AD データコード、入力モード等の取得	

【１】ボード及び本ハンドラーの初期化

int __stdcall AD682_Open_ADSys(HWND Owner, int board_sw);	
引数	Owner:ウィンドーハンドル board_sw:ボード上ロータリースイッチ（SW-BN スイッチ）の設定値
戻り値	正常終了時：ボード ID（0XEH） エラー発生時：エラーコード（負の値）
機能・動作	プラグアンドプレイで設定された各ボードのＩ／Ｏアドレス、及び割り込みレベルを認識すると同時にボードリセット、その他ハンドラー内の各種テーブルやデータバッファ等を初期化する。

【２】サンプリング実行チャンネル関連の設定

int __stdcall AD682_Set_SampCh (int no_ch, int sacn_order[], int range[]);	
引数	no_ch：サンプリングを実行するチャンネル数 scan_order[]：各チャンネルのスキャン順（未使用：固定値） range[]：各チャンネルの入力レンジ番号（未使用：固定値）
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	サンプリングを実行するチャンネル数、入力レンジを設定する。スキャン順序は若いチャンネル番号からの順に行われる。入力レンジ等は後述の電子スイッチにより切り替える事ができる。

【３】サンプリングモードの設定

int __stdcall AD682_Set_SampMode(int trs_trig, int buf_area, int intr_sw);	
引数	trs_trig:AD データ転送の参照フラグ　０：ノットエンプティ遷移、１：ハーフフル遷移 buf_area:ＡＤデータ転送先　０：ＤＬＬバッファ、１：ユーザープログラム内バッファ intr_sw:割り込みイベント発生要因指定 ０：　割り込み不使用
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	buf_area で DLL 管理バッファとした場合は、ハンドラーが自動で DLL バッファに AD データを取り込みユーザープログラム側はステータス取得関数でサンプリング済みデータ点数を認識して専用関数でユーザーバッファへデータを読み出す。ユーザープログラム内バッファとした場合はユーザー側でステータス関数を使用し AD ボード内メモリ使用状況を監視し適宜ユーザープログラムバッファへ AD データを取り込む。

【４】トリガ条件の設定

int __stdcall AD682_Set_Trigger(int trg_mode, int trg_source, int trg_pol, int trg_level);	
引数	trg_mode:サンプリングモード（０：即トリガ、１：ポストトリガ） trg_source:トリガーソース指定（０：ソフトトリガ、１：アナログトリガ、２：デジタルトリガ） trg_pol:トリガ極性指定（０：立ち下がりエッジ、１：立ち上がりエッジ、２：負レベル、３：正レベル） trg_level:アナログトリガレベル（０～２５５）
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	検出されているボードに、トリガ条件を設定する。

【５】レンジトリガ関連の設定

int __stdcall AD682_Set_RangeTrigger(int trg_sel, int trg_level_hi, int trg_level_lo);	
引数	trg_sel:トリガ動作モード（０：アウトレンジ、１：インレンジ、２：デュアルスロープ（＋）、３：デュアルスロープ（－） trg_level_hi:トリガレベル上限値 trg_level_lo:トリガレベル下限値
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	レンジトリガの極性を含む動作モード、レベルの設定を行う。

【６】オプション外部クロック値の設定

int __stdcall AD682_Set_Exclk(long exclk_freq);	
引数	exclk_freq:オプションまたは外部クロックの周波数値（Hz 単位）
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	オプションまたは外部クロックを使用するとき、その周波数値を指定する。この際、Set_Clock 関数を呼び出す前に本関数を実行する必要がある。但し、クロック設定で分周比による設定を行う場合若しくは内部クロックを使用する場合にはこの関数を使用する必要はない。

【７】クロック源及びクロック値の設定

int __stdcall AD682_Set_Clock(int clk_source, int set_mode, int *time_unit, int　*clk_period);	
引数	clk_source:クロック源　０：内部クロック０（１０MHｚ） １：内部クロック１（８．１９２MHｚ） ２：内部クロック２（オプション） ３：外部クロック立下りエッジ ４：外部クロック立上りエッジ set_mode:クロック値の設定方法　０：周期の値 １：分周比 time_unit:クロック周期の単位　０：秒 １：m秒 ２：u 秒 clk_period:クロック周期の値、または分周比
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	ロータリースイッチ番号０のボードに、クロック設定条件を通信する。またスレーブボードには固定の条件を設定する。（外部クロック立ち下がりエッジ、分周比設定、分周比＝１）

【８】サンプリングスタート

int __stdcall AD682_Start_Samp(int no_samp);	
引数	no_samp:1 チャンネル当たりのサンプリングデータ点数
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	サンプリングを実行する。トリガモードがソフトトリガ（即トリガ）である場合は即時サンプリングが開始され、また、何らかのトリガ待ちモードである場合は、その条件が成立するまで待った後サンプリングが開始される。

【９】ボードステータスの取得

int __stdcall AD682_Get_Status(int *sampled, int status[]);	
引数	sampled:現時点でＤＬＬバッファに取り込まれている１チャンネル当たりのサンプリング点数（ポインター渡し） status:ボードの生ステータス
戻り値	正常終了時：０（連続サンプリング実行中）、１（停止中） エラー発生時：エラーコード（負の値）
機能・動作	現時点でのサンプリング済みデータ点数（ＤＬＬバッファ内）及びＡＤボードの生ステータスを取得する。（３－１３項を参照の事）

【10】本ハンドラーの終了

int __stdcall AD682_Close_ADSys(void);	
引数	なし
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	全てのリソースを開放し本ハンドラー動作前の状態に復帰する関数

【11】DLL 管理バッファからのデータ読出し

int __stdcall AD682_Read_DLLData(int data_no, int stno, UINT_PTR *bufptr, int bufsize);	
引数	no_data:チャンネル当たりの読み出しデータ点数 data_pos:読み出すデータの Dll バッファ先頭からのオフセット（チャンネル当たり） bufptr:ユーザーバッファ先頭アドレス（ポインター渡し） buf_size:ユーザーバッファサイズ（バイト単位）
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	Dll バッファ内のデータをユーザーバッファに転送する。

【12】ユーザーメモリへの直接データ転送

int __stdcall AD682_Read_DirectFifo(UINT_PTR *bufptr, int bufsize);	
引数	bufptr:転送先ＡＤデータバッファの先頭アドレス（ポインター渡し） bufsize:転送先ＡＤデータバッファの大きさ（単位：バイト、総データ数の２倍）
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	ＡＤボード上のメモリから直接ユーザープログラム内のデータバッファに転送する。本ボードでは、【６】メモリ使用データ数の取得　によって取得済みデータ数を確認する方法があるため、ソフトウェアサイドから任意のタイミングで上記データ数取得関数を発行することで、その時点で取得可能なデータ数が判明するので、使いやすい関数となっている。

【13】マニュアルサンプリング

int __stdcall AD682_Get_OneScan(UINT_PTR *bufptr, int bufsize);	
引数	bufptr:転送先ＡＤデータバッファの先頭アドレス（ポインター渡し） bufsize:転送先ＡＤデータバッファの大きさ（バイト指定）
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	あらかじめ設定されたチャンネル群に１回のサンプリングを行わせる。

【14】汎用デジタル出力

int __stdcall AD682_Out_Aux(int data);	
引数	data:汎用出力へセットする値（１：ハイレベル、０：ローレベル）
戻り値	正常終了時：　0 エラー発生時：エラーコード（負の値）
機能・動作	SW-BN スイッチの値で指定されるボードについて汎用出力の値を更新する関数

【15】汎用デジタル入力

int __stdcall AD682_Inp_Aux(void);	
引数	board_no:SW-BN スイッチの値で指定されるボード番号
戻り値	正常終了時：汎用入力の値（１：ハイレベル、０：ローレベル） エラー発生時：エラーコード（負の値）
機能・動作	汎用入力の値を取り込む関数

【16】リングバッファ指定有無

int __stdcall AD682_Set_SampLoop(void);	
引数	なし
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	DLL バッファの利用形態を通常モードとするか無限サンプリング用のリングバッファモードとするかを指定する関数

【17】アナログ入力モードの設定

int __stdcall AD682_Set_Inpmode(int inp_line, int code, int resol);	
引数	inp_mode：未使用（アナログ入力信号形式） ad_code：ＡＤデータコード指定（０：バイナリ、１：２の補数） ad_reso：未使用（ＡＤ変換分解能）
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	アナログ変換出力コード指定を行う関数

【18】サンプリングの強制停止

int __stdcall AD682_Stop_Samp(void);	
引数	なし
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	動作中のサンプリングを強制的に停止する関数。この関数は、有限サンプリングを何らかの理由で強制的に停止させたい場合或いは無限サンプリングを行わせている状態で、その動作を停止させたい場合に使用するものである。尚この関数を実行した場合、設定したパラメータ類は保持されている。

【19】残りＡＤデータの読み込み

int __stdcall AD682_Read_RestData(void);	
引数	なし
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	データロスト等のエラー発生時ＡＤボード上の FIFO メモリにデータが残っている場合、本関数でそれを DLL バッファの続きとして読み込むことができる。尚、前記【18】の強制停止を実行すると、残りデータの読み出しはできず、データそのものも無効となる。

【20】各フラグのクリア

int __stdcall AD682_Clear_Flags(int data);	
引数	data：ビット０＝１なら FIFO メモリをクリア ビット２＝１ならエラークリア ビット５＝１なら TIM フラグをクリア ビット６＝１なら INT フラグをクリア ビット７＝１なら EOS フラグをクリア
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	ＡＤボード上の各フラグをクリアする。

【21】外付け・同時サンプルホールド制御

int __stdcall AD682_Set_Shc(int shc);	
引数	shc：外付け同時サンプルホールド制御をするかどうかを指定する変数 ０：しない １：する
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	複数チャンネル同時サンプルホールド回路を使用する場合にサンプリングスキャン開始タイミングを遅らせるかどうかを指定する変数。この変数を“１”とするとサンプルホールド後 1uS 経過した後サンプリングを開始するため、ホールドステップ誤差が（特にチャンネル０について）最小になる。一方、この変数を“０”とすると最高サンプリング速度の低下はなくなる替わりに、ホールドステップ誤差が増加する。

【22】本ハンドラーのバージョン情報取得

int __stdcall AD682_Get_LibVer (int ver);	
引数	ver:取得するバージョンの形態を指定する変数。　０：メジャー＋マイナー番号 １：メジャー番号 ２：マイナー番号
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	本ハンドラーのバージョン番号を取得する関数

FCAD412 独自機能用関数セット

以下の関数は FCAD412PCI に導入されたハードウェアを操作する専用関数群である。

【23】ハーフフルサイズの取得

int __stdcall AD412_Get_HalfSize (void);	
引数	なし
戻り値	正常終了時：Fifo ハーフサイズ値（単位：ワード） エラー発生時：エラーコード（負の値）
機能・動作	システムで検出された本ボード一枚あたりのＦＩＦＯハーフサイズを取得する関数である。この関数を使用することで、現状のＦＩＦＯハーフフルサイズを取得しデータ処理に利用することができる。

【24】メモリ使用データ数の取得

int __stdcall AD412_Get_UsedSize(UINT_PTR *mem_size);	
引数	mem_size:使用済みメモリデータ数を取得するためのバッファ（ポインター渡し）
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	使用済みメモリデータ数を取得する関数

【25】AD 入力モードの設定

int __stdcall AD412_Set_DataCode(int board_no, int *half_full, int *se_diff, int *sence_level, int *uni_bypol, int *range_mode;)	
引数	board_no:対象ボード番号 half_full:FIFO メモリのハーフフルサイズ設定 se_diff:シングルエンド／差動入力切り替え（０：シングルエンド、１：差動入力） sence_level:入力感度切り替え（０：２．５Ｖ、１：５Ｖ、２：１０Ｖ） uni_bypol:ユニポーラ／バイポーラ指定（０：バイポーラ、１：ユニポーラ） range_mode:A レンジ・B レンジ切り替え（０：A レンジ、１：B レンジ）
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	入力モードを指定する関数である。 FCAD412 では、アナログ回路の切り替え機能がメカニカルスイッチではなく半導体による電子切り替え機能となったため、これら回路の設定をアプリケーション側からソフトウェア的に制御する事が可能となった。尚、本関数で設定された内容は一時的なものであり、ボードリセット或いはハードウェアリセット（電源再投入）により、初期状態に復帰する。

【26】AD 入力モードの読出し

int __stdcall AD412_Get_DataCode(int board_no, int *half_full, int *se_diff, int *sence_level, int *uni_bypol, int *range_mode;)	
引数	board_no:対象ボード番号 half_full:FIFO メモリのハーフフルサイズ設定 se_diff:シングルエンド／差動入力切り替え（０：シングルエンド、１：差動入力） sence_level:入力感度切り替え（０：２．５Ｖ、１：５Ｖ、２：１０Ｖ） uni_bypol:ユニポーラ／バイポーラ指定（０：バイポーラ、１：ユニポーラ） range_mode:A レンジ・B レンジ切り替え（０：A レンジ、１：B レンジ）
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	アナログ回路の設定状態を取得する関数である。

４－７　エラーコード

戻り値	不具合の内容	適用関数等
－１	必要な PCI ライブラリが見つからない	AD682_Open_ADSys
－２	FCAD412PCI ボードが見つからなかった	各関数
－３	マスターボードが見つからなかった	各関数
－４	SW-BN スイッチが重複している	AD682_Open_ADSys
－５	データコード指定が間違っている	AD682_Set_Inpmode
－６	指定されたユニット番号のボードが存在しなかった	各関数
－７	指定されたボード数と実際の枚数が異なっていた	AD682_Open_ADSys
－８	オープンされているボードがない	各関数
－９	スタートしているボードがない	各関数
－１０	指定チャンネル数が異常	AD682_Set_SampCh
－１１	転送フラグ指定エラー	AD682_Set_SampMode
－１２	バッファ指定エラー	AD682_Set_SampMode
－１３	トリガ選択エラー	AD682_Set_Trigger 等
－１４	トリガ源選択エラー	AD682_Set_Trigger
－１５	トリガのレベル関係が異常	AD682_Set_RangeTrigger
－１６	外部クロック値設定エラー	AD682_Set_Exclk
－１７	クロックソース指定が異常	AD682_Set_Clock
－１８	クロック周期指定が異常	AD682_Set_Clock
－１９	スキャン周期指定が仕様範囲外	AD682_Set_Clock
－２０	分周比指定が許容範囲外	AD682_Set_Clock
－２１	クロック指定方法が間違っている	AD682_Set_Clock
－２２	Dll バッファポインターがおかしい	AD682_Read_DllData
－２３	Dll バッファサイズがおかしい	AD682_Read_DllData,AD682_Get_OneScan
－２４	何らかの理由で動作していないボードがある	AD682_Get_OneScan
－２６	バージョン番号指定エラー	AD682_Get_Libver
－２９	アナログ設定設定値エラー	AD412_Set_DataCode
－３０	チャンネル数がボード毎に異なっていた	AD682_Start_Samp
－３１	既にサンプリング中	各関数
－３２	サンプリング数が異常だった	AD682_Start_Samp
－３３	ユーザーバッファが指定されている	AD682_Read_RestData
－３４	複数ボードに於いてハーフフルサイズが異なっていた	AD682_Open_ADSys
－３５	データロストが発生していない	AD682_Read_RestData
－３６	メモリ確保に失敗した	ブロックリード命令関連
－３８	外部クロックが設定されていない	AD682_Set_Clock
－３９	シングルモードと差動モードが混在している	AD682_Start_Samp
－４０	サンプルホールドパラメータ異常	AD682_Set_Shc
－４１	SW-BN スイッチの指定が間違っている	AD682_Open_ADSys

第５章　汎用デバイスドライバー仕様書

本仕様書は、フレックスコア製 PCI ボードを制御するために必要となる汎用デバイスドライバーの内容について記述したものです。本デバイスドライバーは、最大で 16 台までの PCI ボードを同時に制御する事が可能ですが、基本的な仕組みは、ドライバーオープン及びドライバースタートから始まり、夫々の PCI ボードリソースを、Ｉ／O制御レジスタへのアクセスにより制御し、全ての制御が終了した段階でドライバーストップ、更にドライバークローズ処理を行って制御全てを終了するところまでを行います。
基本的なリソース制御については、バイト制御、バイトブロック制御、ワード制御、ワードブロック制御、更にダブルワード制御及びダブルワードブロック制御の１２種類（各々リード・ライト処理がある）が定義されています。実際には、本仕様書脱稿の段階でダブルワード制御を行うレジスタを装備したボードは存在しませんが、将来的にそのようなボードが成立する可能性も充分考えられるため、ハンドラー仕様書として内容を整備しています。

本ドライバーは、Windows7 等での PCI ボードインストールとの関係上、対応する専用ドライバーインストール時に共にインストールされるよう設定されています。（即ち専用のインストーラーは存在しません。）

また、本ドライバーは、上記の如く単純なＩ／Ｏレジスタアクセスのみから構成されている為、基本的に無サポートとします。また全てのデータ処理（バイト／ワード／ダブルワード：リード／ライト）については、クリティカルセクションによって排他処理が保証されています。

※　クリティカルセクションについては、特にマルチコアＣＰＵ等の環境で本ドライバーを使用する際に必要となるものですが、ここではその詳細については記述しません。詳細については、マイクロソフト製ビジュアルスタジオ等に詳細な記述があるのでこれを参照してください。

ハンドラー及び関連ファイル名称一覧表

ＯＳ及びビット幅	Windows7　32 ビット	Windows7　64 ビット	WindowsXP　32 ビット
ドライバー名称	PORT7_X86.Dll	PORT7_X64.Dll	PORT7_X86.Dll
インポートライブラリ名称	PORT7_X86.Lib	PORT7_X64.Lib	PORT7_X86.Lib
関数インデックス	PORT7.H	PORT7.H	PORT7.H

上記ドライバー３種類の内、WindowsXP32 ビット用ドライバーについては、システム移行動作確認用としてのみ使用する事を前提として公開しています。このドライバーは、WindowsXP システムで動作していたアプリを Windows7 へ移行する際に、その操作を段階的に（つまり OS の変更を行うこと無しに）新しいドライバーでの動作評価を行うことができるよう用意されているものです。（ハンドラーの内容としては Windows７　３２ビット版と全く同じ構成になっています。）

以下に記述する仕様は、ある固有ベンダーID に従った PCI ボード群に対する汎用デバイスドライバーの仕様書です。ベンダーID は固有の値（現状０ｘ１３ＦＤ）でなければなりません。

【1】ドライバーオープン

int __stdcall Open_Driver(int *device_id);	
引数	device_id:各ボードのデバイス ID をセットする配列（ユーザー側にて確保:最大 16 個）
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	ハンドラー内部の初期化を行い、システム内で検出されたそれぞれのボードのデバイス ID を報告する。

【2】ドライバースタート

int __stdcall Start_Driver(int board_no, int base_no, int *resorce_no);	
引数	board_no: ドライバーオープンで報告された、デバイス ID を格納している配列の要素番号であり、尚且つシステム内で検索され発見された該当ボードの発見順序を表す変数である base_no: ＩＯリソース番号（２固定） resource_no:当汎用ハンドラー内で確定したリソースを指し示す要素番号
戻り値	正常終了時：０ エラー：エラーコード（負の値）
機能・動作	ドライバーオープンで検出されたボード群の内 board_no で指定したボードと通信を開始する。

【3】バイトリード

int __stdcall ReadByte(int resource_no, int offset);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス
戻り値	正常終了時：読み込んだバイト値 エラー発生時：エラーコード（負の値）
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタの値を読み込みユーザーへ返す関数（戻り値＝バイト）

【4】バイトブロックリード

int __stdcall ReadByteBlock(int resource_no, int offset, int count, BYTE *pbuf);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス count:読込みを行うバイト数 pbuf:読み込んだレジスタからの値を保存する先頭アドレス（ポインター）
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタの値を count 回数読み込みユーザーが用意したデータバッファ（pbuf）へ返す関数である。pbuf データバッファのサイズは count（バイト）必要である。

【5】ワードリード

int __stdcall ReadWord(int resource_no, int offset);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス
戻り値	正常終了時：読み込んだワード値 エラー発生時：エラーコード（負の値）
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタの値を読み込みユーザーへ返す関数

【6】ワードブロックリード

int __stdcall ReadWordBlock(int resource_no, int offset, int count, WORD *pbuf);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス count:読込みを行うワード数 pbuf:読み込んだレジスタからの値を保存する先頭アドレス（ポインター）
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	board_no で示されるボードの offset アドレスに存在するレジスタの値を count 回数読み込みユーザーが用意したデータバッファ（ret_value）へ返す関数である。Ret_value データバッファのサイズは count×２（バイト）必要である。

【7】ダブルワードリード

int __stdcall ReadDWord(int resource_no, int offset);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス
戻り値	正常終了時：読み込んだダブルワード値 エラー発生時：エラーコード（負の値）
機能・動作	board_no で示されるボードの offset アドレスに存在するレジスタの値を読み込みユーザーへ返す関数

【8】ダブルワードブロックリード

int __stdcall ReadDWordBlock(int resource_no, int offset, int count, UINT32 *pbuf);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス count:読込みを行うワード数 pbuf:読み込んだレジスタからの値を保存する先頭アドレス（ポインター）
戻り値	正常終了時：０ エラー発生時：エラーコード（負の値）
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタの値を count 回数読み込みユーザーが用意したデータバッファ（pbuf）へ返す関数である。pbuf データバッファのサイズは count×４必要である。

【9】バイトライト

int __stdcall WriteByte(int resource_no, int offset, BYTE data);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス data:レジスタへ書き込む値
戻り値	正常終了時：0 エラー発生時：エラーコード（負の値）
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタへ data を書き込む関数

【10】バイトブロックライト

int __stdcall WriteByteBlock(int resource_no, int offset, int count, BYTE *pbuf);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス count:読込みを行うワード数 pbuf:読み込んだレジスタからの値を保存する先頭アドレス（ポインター）
戻り値	正常終了時：0 エラー発生時：エラーコード（負の値）
機能・動作	board_no で示されるボードの offset アドレスに存在するレジスタへ pbuf バッファから count 回数書き込みを行う関数である。Pbuf データバッファのサイズは count（バイト）必要である。

【11】ワードライト

int __stdcall WriteWord(int resource_no, int offset, WORD data);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス data:レジスタへ書き込む値
戻り値	正常終了時：0 エラー発生時：エラーコード（負の値）
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタへ data に保存されているデータを書き込む関数

【12】ワードブロックライト

int __stdcall WriteWordBlock(int resource_no, int offset, int count, WORD *pbuf);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス count:書き込みデータ数 pbuf:レジスタへ書き込む値を保存しているアドレス（ポインター）
戻り値	正常終了時：0 エラー発生時：エラーコード（負の値）
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタへ pbuf に保存されているデータを書き込む関数。pbuf データバッファのサイズは count×２（バイト）必要である。

【13】ダブルワードライト

int __stdcall WriteDWord(int resource_no, int offset, UINT32 data);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス data:レジスタへ書き込む値
戻り値	正常終了時：0 エラー発生時：エラーコード（負の値）
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタへ data に保存されているデータを書き込む関数

【14】ダブルワードブロックライト

int __stdcall WriteDWordBlock(int resource_no, int offset, int count, UINT32 *pbuf);	
引数	resource_no:スタートドライバー関数から返されたボードリソースインデックス offset:操作対象レジスタのオフセットアドレス count:書き込みデータ数 pbuf:レジスタへ書き込む値を保存しているアドレス（ポインター）
戻り値	正常終了時：0 エラー発生時：エラーコード（負の値）
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタへ pbuf に保存されているデータを書き込む関数。pbuf データバッファのサイズは count×４（バイト）必要である。

【15】ドライバーストップ

int __stdcall Stop_Driver(int resource_no);	
引数	resource_no: ドライバースタート関数から返されたボードリソースインデックス
戻り値	正常終了時：0 エラー発生時：エラーコード（負の値）
機能・動作	全てのユニットで使用していたメモリリソースを開放し通信を終了する。

【16】ドライバークローズ

int __stdcall Close_Driver(void);	
引数	なし
戻り値	正常終了時：0 エラー発生時：エラーコード（負の値）
機能・動作	全てのボードについてハンドルを開放する。

エラーコード

戻り値	内容
－1	必要な PCI ライブラリが見つからない
－2	ライブラリのデバッグオプションが無効だった
－3	PCI ボードが見つからなかった
－4	ボード番号指定がシステム上の枚数を上回っている
－5	オープンしていないボードをクローズしようとした
－6	定義されていないベースナンバーを参照しようとした
－7	オープンしていないボードに対してアクセスが行われた
－8	リソース範囲不適
－9	内部メモリ確保エラー

第６章　保守、その他

６－１　故障・トラブル等の原因と対処

本ボードは、ＤＯＳ／Ｖパソコンで全数検査の上出荷されています。
お手元での動作確認方法は１－６項に記されています。動作に不具合がある時は、以下の諸点を再点検して下さい。
それでも不明なときは、システム構成等をご記入の上、弊社品質管理部まで御連絡下さい。情報伝達手段としてfax及びEmailを準備しておりますが、ご使用しやすい方の手段をお選びの上、御連絡願います。
一般的に言って、応答速度を求められるときはEmailが、複雑な情報を含む場合はfaxがお勧めだと思います。

再点検、確認ポイント

ボードID

単体で動作させる時、この番号が０以外の設定になっているとハンドラーがユニットを認識しません。実際に起こり得る局面としては、マスタースレーブ接続で使用されていたスレーブユニットを、単独で使用する等の局面が考えられます。

トリガ方法

アナログレベルトリガは、アナログ変換値の値で、アナログエッジトリガは、アナログ変換値の値の変化で、夫々トリガを認識します。そのため、アナログレベルトリガでは、サンプリング開始と同時にトリガ検出が行われることがある一方で、アナログエッジトリガでは、アナログ電圧値がスレッショルドをよぎる変化を起こさないとトリガとして認識しません。詳細については、３－６項を参照して下さい。

デジタル入出力

本ボードのデジタル入力に接続できる信号源はTTL(LS,CMOS等の５Ｖ電源動作素子)に限ります。
それ以外の電源電圧を用いている信号源を接続すると、殆ど一瞬のうちに入力素子が破壊します。
出力についてもTTL信号源以外の信号源に接続することは、即出力素子の破壊を招きます。
本ボードで使用しているデジタル入出力素子の絶対最大定格は－０．６Ｖ～＋７．０Ｖとなっています。

アナログ入力

本ボードの過電圧入力保護は±３５Vが最大です。そのため一瞬であってもこの電圧範囲を超える電圧が印加されるとアナログ入力回路素子の破壊につながります。また、アナログ回路は、デジタル回路と異なり異常電圧が伝播していく可能性が高いため、異常電圧による被害はデジタル回路に比べて大きくなりがちです。

複数チャンネル使用時の各信号源のグランド電位差にも注意が必要です。特に様々な機器からの入力を接続する機会が多いこともあいまって、この原因による回路破壊も時折見られます。予防策としては、夫々の回路を接続する前に、機器間のグランド電位差をテスターなどで確認しておく事があげられます。一般的には交流的な電位差がある場合が多いと考えられますが、直流的な電位差についても確認が必要です。

更に、本ボードのグランド電位は登載されているパソコンのグランド電位と共通であるという点にも注意が必要です。

６－２　修理のときは

本製品使用上の質問・トラブル対応・故障修理などは入手経路の如何にかかわらず、弊社宛に直接御相談下さい。
商社等を経由されても弊社として問題はありませんが、情報交換の密度、修理完了までの時間などの面で多少不利になるのではないかと思われます。
また、導入当初からの不具合については、誤解や情報不足による事が多いので、事前にご相談下さい。
無償修理と有償修理の場合分けにつきましては以下の通りとなります。
納入後１年間は自然故障、及び弊社製造上の問題に起因した事が明らかな故障製品に対して無償修理を行います。

但し保証期間中であっても、次の場合は有償修理となります。

ア　取扱上の不注意、誤用による故障および損傷
イ　弊社以外での修理、改造、分解掃除等による故障および損傷
ウ　泥、砂、水などのかぶり、落下、衝撃等が原因で発生した故障および損傷
エ　火災、地震、水害、落雷その他の天変地異、公害や異常電圧による故障および損傷
オ　保管上の不備（高温多湿の場所等）や手入れの不備による故障
カ　接続している他の機器に起因して故障が生じた場合
キ　その他使用者側の責に帰する故障

また、不良要因が再現されなかった場合につきましては、着払いにより製品を返却させて頂く場合もありますのでご了承下さい。

修理は宅配便によるセンドバックで行います。尚運賃は互いに発送する側が負担するものとします。
出張修理は行っておりません。簡単な故障であれば一週間程度で修理可能ですが、故障状況によっては更に日数を要する事もあります。
また、有償修理の場合の修理料金については、簡単な故障で基本料金￥１０，０００円＋部品代とお考え下さい（事務手数料を含む）。修理費用限度額がある場合は、お申し出頂ければ超過する事が明確になった時点で御連絡いたします。

修理品送付先
〒301-0853
茨城県龍ヶ崎市松ヶ丘３丁目１８番地３
フレックスコア
品質管理部　林
E-mail: support@flexcore.jp
Fax:050-3488-3354

６－３　アナログ入力範囲の再調整

１－６項に示すような形で動作確認を行い、入力範囲やオフセットの変化が認められるときは再調整が必要です。 アナログ回路は経年環境変化に対する保守を定期的に行うことが望ましく、夏冬の使用環境・周囲温度に差がある場合は季節単位、通年安定した使用環境の場合は１～２年に１度は校正することが理想的です。

再調整の方法・手順を以下に記しますが、被調整対象のＦＣＡＤ１０８ＳＰＣＩ－１６MWの他に、高確度の基準電源と高精度の電圧計、各入力を個別に切り替える事ができるスイッチボックス及びWindow XP／Vistaがインストールされたパソコンを準備する必要があります。

６－４　外部制御信号、タイミング等